論 文

Technical Papers

多頻度グラフパターンの完全な高速マイニング手法

Fast and Complete Mining Method for Frequent Graph Patterns

猪口 明博 Akihiro Inokuchi
大阪大学産業科学研究所
The Institute of Scientific and Industrial Research, Osaka University, 8-1 Mihogaoka, Ibaraki, Osaka 567-0047
inokuchi@ar.sanken.osaka-u.ac.jp, http://www.ar.sanken.osaka-u.ac.jp/inokprjp.html

鷲尾 隆 Takashi Washio
（同　上）
washio@ar.sanken.osaka-u.ac.jp, http://www.ar.sanken.osaka-u.ac.jp/washprjp.html

元田 浩 Hiroshi Motoda
（同　上）
motoda@ar.sanken.osaka-u.ac.jp, http://www.ar.sanken.osaka-u.ac.jp/motoprjp.html

熊澤 公平 Kohei Kumazawa
株式会社 リクルート
RECRUIT CO.,LTD. Recruit Ginza 8th Bldg, 8-4-17 Ginza Chuo-ku, Tokyo 104-8001
kumazawa@isr.recruit.co.jp

荒井 尚英 Naohide Arai
（同　上）
arai@in.recruit.co.jp

Keywords: Graph Structured Data, Frequent Graph, Normal Form, Canonical Form

Summary

In the field of data mining, much attention has been paid to the Basket Analysis. Basket Analysis derives frequent itemsets from database, but its ability of mining is limited to transaction data consisting of items. Some approaches to discover characteristic graph structure from a set of given graph structured data have been researched in the field of inductive machine learning. However, most of them are not suitable for the applications which require to search all frequent graph patterns in the large amount of data. In this paper, we propose a novel principle and its algorithm that derives the characteristic patterns which frequently appear in massive graph structured data. Our algorithm can derive all frequent patterns from both the directed and undirected graph structured data having loops, multiple types of nodes and links.

1. はじめに

膨大なデータの中から有用な，あるいは興味のあるパターンを明示的な知識として発掘しようとするデータマイニングは，計算機能力を最大限に活用したアプローチである．パターンの有用性，新規性，興味深さなどは領域（事前）知識に依存するため，一般的な評価を行う指標を考えることは困難である．しかし，一般に多くの事例を説明する知識は有用であると考えられるので頻度と確信度を指標として有用な知識を取り出すことは可能である [元田 99]．そのためデータマイニングの分野においてはデータ間の共起事象を相関規則として求めるバスケット分析が主流の 1 つとなっている．Apriori アルゴリズム [Agrawal94] を用いるバスケット分析はアイテム間の共起相関を高速に抽出できるが，一方でアイテムで表現されるデータ構造には制限や限界があるので，バスケット分析は基本的に集合からなるデータベース中からしか相関ルールを取り出せない．時系列データを取り扱えるように拡張した研究 [Arimura 98, Shintani 98] も報告されているが，時系列ではより複雑なデータ構造を表現することは困難である．

このようにより複雑なデータ構造（関係）を表現するにはより強力な言語が必要である．現在知られている手法では一階述語論理を用いた帰納論理プログラミング (ILP) が強力であろう．この手法は探索空間が非常に広いので，表現力に制限したり，探索制御にヒューリスティックスを用いるなどの対策が必要である．一方，関係の中にも述語論理ほど強力な表現力がなくても表せるものは多い．多くの事例はグラフ構造を用いて表現できる．機械学習の分野ではグラフ構造データを扱う手法があり [Yoshida 95, Motoda 97, 前原 99]，いずれの手法も与えられたデータから特徴的パターンを取り出す手法である．しかし，計算時間の観点からいずれの手法も Greedy 探索を戦略としており，膨大なデータを対象として，完全な探索を行うことを目標とした研究は筆者等の知るかぎりない．従って，バスケット分析のように計算時間を考慮し

た上で完全探索を行い，従来のバスケット分析では扱えない構造を持つデータを解析できれば，これまでとは違う知識がデータベースから得られる可能性がある．

そこで本稿では膨大なデータから頻繁に出現する特徴的グラフパターンを全て抽出するアルゴリズムを提案する．我々のアルゴリズムは自己ループを含み，ノード及びリンクが複数種類（ラベル）を持つ有向・無向グラフから特徴的パターンを抽出することが可能である．またこのアルゴリズムは互いに連続せず分離した位置にある特徴的パターン抽出，リンクやノードに全く種類がないグラフのトポロジーのみ与えられたデータからの特徴的パターン抽出も可能である．

2 章で従来のバスケット分析手法の概略を説明し，3 章で Apriori アルゴリズムのグラフ構造データへの拡張について述べる．4 章でテストデータによる評価実験結果について述べ，5 章で提案手法の応用例をあげる．6 章では本研究と関連のある研究を紹介する．

2. バスケット分析

2·1 バスケット分析

バスケット分析では，1 つ以上のアイテム（item）の集合であるトランザクション（transaction）で構成されるデータベースを対象する．例えば，コンビニエンスストアで各顧客が購入した商品の情報は以下のようにデータベースに保存されている．

$$\text{顧客}_1 : \{milk, bread, butter, ---\}$$
$$-----$$
$$\text{顧客}_n : \{milk, bread, apple, ---\}$$

バスケット分析の第 1 段階は支持度（support）を基にデータベースから多頻度アイテム集合を導出することである．アイテム集合 I の支持度は出現頻度を表し，全トランザクションの中でそのアイテム集合を含むものの割合を示す値である．即ち，

$$sup(I) = \frac{I \text{を含むトランザクション数}}{\text{全トランザクション数}}$$

である．バスケット分析では，最小支持度（minimum support）という閾値を設定し，それより高い支持度を持つアイテム集合を取り出す．最小支持度以上の支持度を有するアイテム集合を多頻度アイテム集合（frequent itemset）と呼ぶ．

2·2 Apriori アルゴリズム

Apriori アルゴリズムは大規模データから多頻度アイテム集合を効率的に取り出すアルゴリズムである．このアルゴリズム構造は比較的単純でメモリ消費量が少なく，かつ効率が高い特徴を有する．図 1に Apriori アルゴリズムの概略を示す． ここで F_k は要素数 k の多頻度アイテム集合の集合，C_k は多頻度アイテム集合の候補の集合

```
1) F_1 = { Frequent itemsets of cardinality=1} ;
2) for(k = 1; F_k ≠ ∅; k++) do begin
3)    C_{k+1} =apriori-gen(F_k); //New candidates
4)    forall transactions t ∈ Database do begin
5)       C'_t =subset(C_{k+1}, t); //Candidates contained in t
6)       forall candidate c ∈ C'_t do
7)          c.count++;
8)    end
9)    F_{k+1} = { c ∈ C_{k+1} | c.count≥ minimum support }
10) end
11) Answer= ⋃_k F_k ;
```

図 1 Apriori アルゴリズム

である．

関数 Apriori-gen は，join 部と prune 部からなる．join 部では要素数 1 の多頻度アイテム集合からはじめて，ボトムアップ的に要素数 k の多頻度アイテム集合から要素数 $k+1$ の多頻度アイテム集合の候補を作り出すことを行う．具体的に述べると，アイテムは例えば辞書順のような順序関係を持つ，$k-1$ 個の要素が共通な要素数 k の 2 つの多頻度アイテム集合

$$P_k = \{item_1, item_2, \cdots, item_{k-1}, item_k\}$$
$$Q_k = \{item_1, item_2, \cdots, item_{k-1}, item'_k\}$$
$$\text{但し，} item_1 < item_2 < \cdots < item_k < item'_k$$

より，要素数 $k+1$ の多頻度アイテム集合の候補を上記の 2 つのアイテム集合の和集合

$$\begin{aligned} R_{k+1} &= P_k \cup Q_k \\ &= \{item_1, \cdots, item_{k-1}, item_k, item'_k\} \end{aligned}$$

で生成する．prune 部では，さらに候補を絞り込むために，このようにして候補にあがった要素数 $k+1$ のアイテム集合に対して，要素数 k のアイテム集合からなる各部分集合が全て多頻度アイテム集合として存在しているかどうかをチェックする．これは多頻度アイテム集合であるためには，その部分集合は全て多頻度アイテム集合でなければならないためである．要素数 k の多頻度アイテム集合は前段階の処理で全て取り出されており既知であるので，効率的にチェックすることが可能である．図 1の関数 subset では，トランザクション t に含まれる全ての多頻度アイテム集合の候補を見つけ出す．

このように多頻度アイテム集合の候補についてのみ，実際の支持度をデータから検索して頻度を計算する．最小支持度を越えるアイテム集合の要素数の最大を k_{max} とすると，この Apriori アルゴリズムにより高々$k_{max}+1$ 回しかデータベースにアクセスしないために効率的に全ての多頻度アイテム集合を求めることができる．

3. グラフ構造データへの拡張

前節では従来のバスケット分析について述べたが，本節ではグラフ構造データを扱うために Apriori アルゴリズムのグラフ構造データへの拡張を提案する．初めに，リ

ンクに種類があるグラフ，あるいは自己ループを持つグラフに前処理を施すことによって，それらを持たないグラフに帰着する手法を述べる．また本稿では，代数演算を利用することができるので，隣接行列を用いてグラフ構造を表現する．しかし，隣接行列の行（列）を入れ替えることによって生成される隣接行列もまた同型なグラフを表している．同型なグラフの隣接行列の記述方法は様々であり，それによって探索空間も膨大になる．そこで，探索空間を制限するための正規形と同型性を判定するための正準形の概念を導入する．

3·1 グラフ構造データの表現

本稿では 1 つのグラフを 1 つのトランザクションと考え，多くのトランザクションに含まれる多頻度パターンを効率的に抽出するアルゴリズムを提案する．例えば，化学構造データの場合，化合物それぞれがトランザクションに相当し，このようなデータから特徴的パターンを取り出すと，ベンゼン環やメチル基などが取り出されるであろう．グラフ構造データは隣接行列を用いて表すことが可能である．グラフのノードの数をグラフの大きさ，隣接行列の i 行及び i 列に相当するノードを第 i ノードとし，v_i と表す．大きさ k のグラフの隣接行列を X_k と表し，その要素を x_{ij}，グラフを $G(X_k)$ で表す．ノードの種類を N_p とし，リンクの種類を L_q とする．最小支持度は以下のように定義する．

$$sup(G) = \frac{TrN(G)}{\text{全トランザクションの数}}$$

ここで，$TrN(G)$ はグラフ G を誘導部分グラフとして含むトランザクションの数である．誘導部分グラフ（induced subgraph）とはグラフ G' をグラフ G の部分グラフとし，$V(G), E(G)$ をそれぞれグラフ G のノードの集合，リンクの集合とするとき，任意のノード $u, v \in V(G')$ に対して，$\{u,v\} \in E\{G\} \Leftrightarrow \{u,v\} \in E\{G'\}$ が成り立つとき，G' を G の誘導部分グラフという．最小支持度は従来のバスケット分析と同様に定義し，最小支持度を超える支持度を有するグラフを多頻度グラフ（frequent graph）と呼ぶ．

本研究で対象とするデータはノード及びリンクに種類があり，自己ループを持つ無向・有向グラフである．もし対象とするグラフ構造データのリンクに種類がある場合，あるいは自己ループを持つ場合には，簡単な前処理を行い，ノードのみに種類が存在し，自己ループを持たないグラフ構造に変換する．具体的に述べると，図 2(a) は N_1，N_2 のラベルを持つ 3 つのノードから構成されるグラフであり，N_1 のラベルを持つノードの 1 つは自己ループを持つ．L_1，L_2 はリンクの種類である．このようにグラフ構造データのリンクに種類がある場合は，図 2(b) のようにリンクの種類が L_1 であるリンクを N_2 から N_1 への種類を持たない直接のリンクとノードの種類が L_1 であるノード，及び N_2 から L_1 と L_1 から N_1 と

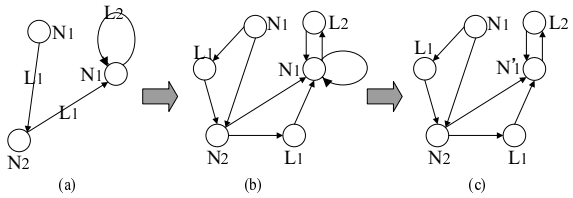

図 2 前処理

いう 2 つのリンクに置き換える．図 2(a)，(b) のグラフ構造を隣接行列で表すとそれぞれ隣接行列 (1)，(2)のようになる．

$$\begin{array}{c} \\ N_1 \\ N_1 \\ N_2 \end{array} \begin{array}{c} \begin{array}{ccc} N_1 & N_1 & N_2 \end{array} \\ \begin{pmatrix} L_2 & 0 & 0 \\ 0 & 0 & L_1 \\ L_1 & 0 & 0 \end{pmatrix} \end{array} \tag{1}$$

$$\begin{array}{c} \\ N_1 \\ N_1 \\ N_2 \\ L_1 \\ L_1 \\ L_2 \end{array} \begin{array}{c} \begin{array}{cccccc} N_1 & N_1 & N_2 & L_1 & L_1 & L_2 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{2}$$

N_2 から N_1 への直接のリンクの必要性を図 3を用いて説明する．graph2 を誘導部分グラフとして持たない graph1 に対し以上で述べた前処理を行うと，graph1-1 のように変換され，graph1-1 は graph2 を誘導部分グラフとして持たない．これに対し，直接のリンクを張らない処理をすると graph1-2 のように変換され，graph1-2 は誘導部分グラフとして graph2 を持つことになる．このような前処理すると，本来の支持度とは違った支持度を計算することになるからである．

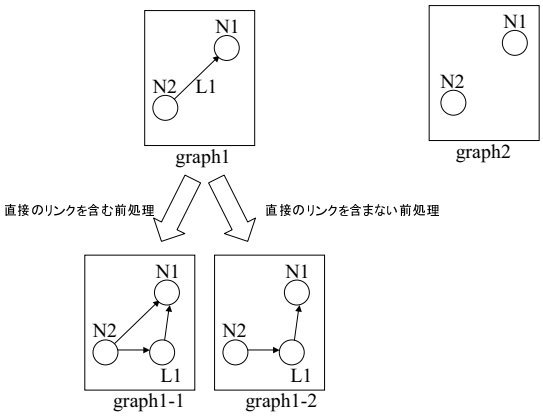

図 3 直接のリンクの必要性

さらにグラフ構造データが自己ループを含む場合には，自己ループを持つノードの種類を別の種類として扱う．図

2(c) は図 2(b) をノードの種類を置き換えた例であり，図 2(c) を隣接行列で表すと

$$\begin{array}{c} \\ N_1' \\ N_1 \\ N_2 \\ L_1 \\ L_1 \\ L_2 \end{array} \begin{array}{c} \begin{array}{cccccc} N_1' & N_1 & N_2 & L_1 & L_1 & L_2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{3}$$

となる．ただし，以上の処理を行うと N_1 と N_1' が同一のラベルが付いていたという情報が失われるため，アルゴリズム終了時に N_1' を自己ループを持つ N_1 に戻す操作を行う．

これによりリンクに種類があるグラフ，または自己ループを持つグラフは前処理によってリンクに種類がなく，自己ループを持たないグラフに帰着できる．これ以降ではグラフのノードのみが種類も持ち自己ループがないグラフについて述べる．

グラフ構造は隣接行列で表現されるが，どのノードを隣接行列の i 行（i 列）にするかによって隣接行列の表現が変わる．そこで，以下のようにグラフ構造のノードの種類間に例えば辞書順のような順序関係を設ける．

$$N_1 < N_2 < \cdots < N_p < \cdots$$

あるグラフ構造データのノードで種類の順序が最も若いノードを第 1 ノードとし，順に第 2 ノード，第 3 ノードとする．例えば図 2(c) のグラフを隣接行列で表現すると，

$$\begin{array}{c} \\ N_1 \\ N_2 \\ N_1' \\ L_1 \\ L_1 \\ L_2 \end{array} \begin{array}{c} \begin{array}{cccccc} N_1 & N_2 & N_1' & L_1 & L_1 & L_2 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{4}$$

となる．ただし

$$N_1 < N_2 < N_1' < L_1 < L_2$$

とする．

以上では有向グラフの表現方法について述べてきたが，無向グラフの場合も同様の前処理を行い，ノードのみが種類を持ち，自己ループを持たないグラフに変換する．ただし，無向グラフの場合には有向グラフと違い対称行列となる．図 4を隣接行列で表現すると以下のようになる．

$$\begin{array}{c} \\ N_1 \\ N_2 \\ N_3 \\ N_3 \\ N_4 \end{array} \begin{array}{c} \begin{array}{ccccc} N_1 & N_2 & N_3 & N_3 & N_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{5}$$

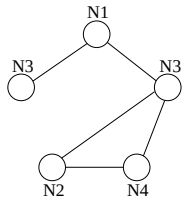

図 4 無向グラフ

次に，無向グラフの隣接行列 X_k について，対角要素を除く上三角要素に注目し，式 (7)のように隣接行列のコード $code(X_k)$ を 2 進数で表す．

$$X_k = \begin{pmatrix} 0 & x_{1,2} & x_{1,3} & \cdots & x_{1,k} \\ x_{2,1} & 0 & x_{2,3} & \cdots & x_{2,k} \\ x_{3,1} & x_{3,2} & 0 & \cdots & x_{3,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{k,1} & x_{k,2} & x_{k,3} & \cdots & 0 \end{pmatrix} \tag{6}$$

$$code(X_k) = x_{1,2}x_{1,3}x_{2,3}x_{1,4}x_{2,4}\cdots x_{k-1,k} \tag{7}$$

一方，有向グラフの場合には，隣接行列をコードを 4 進数で表す．隣接行列 (6)に対し，$(x_{i,j}, x_{j,i}) = (0,0)(i<j)$ ならば符号 0 を割り当てる．同様に (1,0),(0,1),(1,1) ならばそれぞれ 1,2,3 を割り当てる．すなわち，

$$code(X_k) = c_1 c_2 c_3 \cdots c_{\frac{k(k-1)}{2}} \tag{8}$$

$$c_{\frac{j(j-1)}{2}-(j-i-1)} = 2x_{j,i} + x_{i,j} \quad (i<j) \tag{9}$$

とする．例えば，隣接行列 (4)は

$$code(X_k) = 101012120000300 \tag{10}$$

となる．

3·2 多頻度グラフ抽出アルゴリズム

§1 多頻度グラフの候補生成

従来の Apriori ではアイテム間に順序関係を設け，効率的に多頻度アイテム集合を生成する．要素数 k の多頻度アイテム集合から要素数 $k+1$ の多頻度アイテムの候補を生成する際には，ある条件を満たす場合にのみ和集合を計算し多頻度アイテム集合の候補を生成した．

グラフ構造データに対する拡張においても，以下に述べるように条件を設定し，その条件を満たす場合にのみ 2 つのグラフを結合し，サイズの大きな多頻度グラフの候補を順次生成していく．

条件 1 グラフの大きさが k の多頻度グラフを 2 つ考え，その隣接行列を X_k, Y_k とする．X_k, Y_k の k 行及び k 列以外の要素が全て等しいとき，すなわち各々第 k ノードを除いて構造が等しいとき，以下のように X_k, Y_k を結合し，Z_{k+1} 生成する．

$$X_k = \begin{pmatrix} X_{k-1} & \boldsymbol{x}_1 \\ \boldsymbol{x}_2^T & 0 \end{pmatrix}$$

$$Y_k = \begin{pmatrix} X_{k-1} & \boldsymbol{y}_1 \\ \boldsymbol{y}_2^T & 0 \end{pmatrix}$$

$$Z_{k+1} = \begin{pmatrix} X_{k-1} & \boldsymbol{x}_1 & \boldsymbol{y}_1 \\ \boldsymbol{x}_2^T & 0 & z_{k,k+1} \\ \boldsymbol{y}_2^T & z_{k+1,k} & 0 \end{pmatrix} = \left(\begin{array}{cc|c} \multicolumn{2}{c|}{\multirow{2}{*}{X_k}} & \boldsymbol{y}_1 \\ & & z_{k,k+1} \\ \hline \boldsymbol{y}_2^T & z_{k+1,k} & 0 \end{array}\right)$$

ここで，X_{k-1} は大きさ $k-1$ のグラフの隣接行列，$\boldsymbol{x}_i, \boldsymbol{y}_i (i=1,2)$ は $(k-1)\times 1$ の縦ベクトルである．

条件 2 また隣接行列 X_k の第 i ノードのノードの種類を $N(X_k,i)$ とすると隣接行列 X_k, Y_k, Z_{k+1} の間には以下のような関係がある．

$$N(X_k,i) = N(Y_k,i) = N(Z_{k+1},i)$$
$$N(X_k,i) \leq N(X_k,i+1)$$
$$i = 1,\cdots,k-1$$
$$N(X_k,k) = N(Z_{k+1},k)$$
$$N(Y_k,k) = N(Z_{k+1},k+1)$$
$$N(X_k,k) \leq N(Y_k,k)$$

ただし，隣接行列 Z_{k+1} の $(k,k+1)$ 要素，$(k+1,k)$ 要素を X_k, Y_k から決定することはできない．Z_{k+1} のグラフ構造の可能性として，無向グラフの場合，$G(Z_{k+1})$ の第 k ノードと第 $k+1$ の間にリンクが存在する場合とリンクが存在しない場合が考えられる．そこで $(k,k+1)$ 要素，$(k+1,k)$ 要素が共に 0 か 1 の 2 つの隣接行列が生成される．一方，有向グラフの場合には第 k ノードから第 $k+1$ ノードにリンクが存在する場合，その逆方向に存在する場合，双方向のリンクが存在する場合とリンクが存在しない場合の 4 つの可能性がある．そこで有向グラフの場合には 4 つの隣接行列を生成する．隣接行列 X_k, Y_k をそれぞれ Z_{k+1} の第 1 生成行列，第 2 生成行列と呼ぶ．

ここで，2 つの隣接行列を結合させてできた隣接行列 Z_{k+1} の第 1 生成行列，第 2 生成行列の第 k ノードのノードの種類が等しい場合を考える．上記の X_k を第 1 生成行列，Y_k を第 2 生成行列とした場合と，逆に X_k を第 2 生成行列，Y_k を第 1 生成行列とした場合では後者の場合が冗長である．そこで，このような冗長な生成を避けるため，以下の関係にある時のみ結合を行う．

条件 3

$$code(\text{第 1 生成行列}) \leq code(\text{第 2 生成行列})$$

以上のような条件のもとで生成される隣接行列を**正規形**（normal form）と呼び，非正規形の隣接行列は生成しない．以上が従来の Apriori アルゴリズムの関数 Apriori-gen の join 部に相当する．

次に従来の Apriori アルゴリズムの関数 Apriori-gen の prune 部に相当する部分について述べる．結合してできたグラフ $G(Z_{k+1})$ の第 i ノード $(1 \leq i \leq k-1)$ を開放除去したグラフの隣接行列が全て多頻度グラフを表す隣接行列であれば，それを多頻度グラフの候補と考えられる．ここで，第 i ノードの開放除去とは，第 i ノードとそれにつながるリンクを全て除き，大きさが k の誘導部分グラフを得る操作である．先にも述べたように，このアルゴリズムでは正規形の隣接行列しか探索生成しないために，第 i ノードを開放除去したグラフの隣接行列が正規形でなければ，それが多頻度グラフであるかを過去の探索から容易にチェックする事ができない．よって，非正規形の隣接行列を正規化する手法が必要である．

正規化の具体例を図 5の非正規形の隣接行列 X_4 について示す．図 5の $G(X_4)$ のノードは種類を持たないものとし，隣接行列の下の数字はその隣接行列のコードである．v_i は正規化される隣接行列 X_4 の第 i ノードである．(A) はじめにノード数が 1 からなる X_4 の部分グラフの隣接行列を考える．(B) 多数ある正規形の中で，最終的に 1 つ正規形が見つかれば十分なので，結合の組み合わせを限定し，v_1 を元にして結合を行う．(C) 結合により得られない情報，例えば，v_1, v_2 からなる隣接行列の (1,2) 要素，(2,1) 要素は元の隣接行列 X_4 の x_{12} 及び x_{21} から補う．このため，多頻度グラフの候補を計算する際には 2 個（有向グラフの場合には 4 個）の隣接行列が生成されるが，この場合 1 つの隣接行列だけについて計算する．(D) 次にノード数が 2 の隣接行列の結合を行う．(E) このとき正規形の条件から，隣接行列のコードが最小の行列を第 1 生成行列とする．コードが同じ隣接行列が複数ある場合にはどれか 1 つを選択する．以下，順に繰り返し，非正規形の隣接行列 X_4 を再構築すれば正規化された隣接行列を得る．これにより開放除去して得られた全ての誘導部分グラフが過去の探索結果から多頻度グラフであることを確定できれば $G(Z_{k+1})$ は多頻度グラフの候補となる．

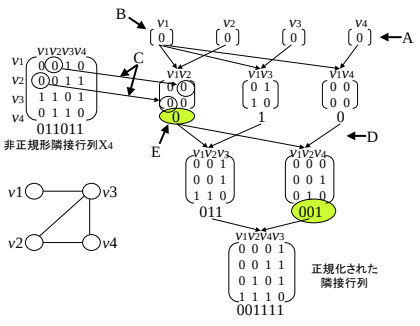

図 5 正規化の例

§2 正準形

3・2・1 節では多頻度グラフの候補の生成方法について述べたが，正規形の中には同じグラフを表す隣接行列が存在する場合がある．以下の 2 つの隣接行列がその例である．

$$X_3 = \begin{array}{c} \\ N_1 \\ N_1 \\ N_1 \end{array} \begin{array}{c} \begin{array}{ccc} N_1 & N_1 & N_1 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

$$Y_3 = \begin{array}{c} \\ N_1 \\ N_1 \\ N_1 \end{array} \begin{array}{c} \begin{array}{ccc} N_1 & N_1 & N_1 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \end{array}$$

大きさが 3 より大きいトランザクションから大きさが 3 の誘導部分グラフを取り出し，あるトランザクションの時は X_3 のカウンタを増やし，別の時は Y_3 のカウンタを増やしたのでは正確な頻度を知ることはできない．頻度の計算の前に，どの隣接行列が同じグラフを表す行列であるかを知る必要がある．そこで，同じグラフを表現する正規形の隣接行列のうちコードが最小のものを**正準形**（canonical form）とし，図 6に示すアルゴリズムにより正準形とそれへの変換行列を求める．変換行列とは，一般に以下のように定義する．隣接行列 (3)と隣接行列 (4)の関係のように，2 つの隣接行列 X_k と Y_k のグラフ構造が等しいとき，X_k から Y_k への変換行列 W_k を考え，式 (11)のように定義する．

$$w_{ij} = \begin{cases} 1 & X_k \text{ の第 } i \text{ ノードが } Y_k \text{ の} \\ & \text{第 } j \text{ ノードに相当する時} \\ 0 & \text{その他} \end{cases} \qquad (11)$$

Y_k は隣接行列 X_k と変換行列 W_k を用いて

$$Y_k = W_k^T X_k W_k$$

と表される．

次に図 6の正準化アルゴリズムについて説明する．大きさが k の隣接行列 X_k の正準形とそれへの変換行列を探す際に，大きさが $k-1$ の多頻度グラフの正準形への変換行列は全て既知でありその 1 つをメモリに記憶しておく．$G(X_k)$ の第 m ノード $(1 \leq m \leq k)$ を開放除去し，その隣接行列を正規化する．正規形への変換行列を $T^m_{k-1}(X_k)$ とする．正規化された隣接行列の正準形への変換行列を $S^m_{k-1}(X_k)$ とする．X_k の変換行列 $S^m_k(X_k), T^m_k(X_k)$ を $S^m_{k-1}(X_k), T^m_{k-1}(X_k)$ から式 (12)，式 (13)のように生成する．

$$s_{ij} = \begin{cases} s^m_{ij} & 0 \leq i \leq k-1 \\ & \quad \text{かつ } 0 \leq j \leq k-1 \\ 1 & i = k \text{ かつ } j = k \\ 0 & \text{その他} \end{cases} \qquad (12)$$

$$t_{ij} = \begin{cases} t^m_{ij} & i < m \text{ かつ } j \neq k \\ t^m_{i-1,j} & i > m \text{ かつ } j \neq k \\ 1 & i = m \text{ かつ } j = k \\ 0 & \text{その他} \end{cases} \qquad (13)$$

ここで，$s^m_{ij}, s_{ij}, t^m_{ij}, t_{ij}$ はそれぞれ $S^m_{k-1}, S^m_k, T^m_{k-1}, T^m_k$ の要素である．X_k に対する正準形は図 6のアルゴリズムで求められる．X_k の正準形のコードは基本的に

$$\min_{m=1,\cdots,k} code((T^m_k S^m_k)^T X_k (T^m_k S^m_k)) \qquad (14)$$

で与えられ，正準形への変換行列は式 (14)を最小にする $T^m_k S^m_k$ である．しかし，式 (14)の計算途中で正準形への変換行列 S'_k が既知である隣接行列 $(T^m_k S^m_k)^T X_k (T^m_k S^m_k)$ が見つかれば，X_k の正準形は式 (15)のように与えられ，全ての m について計算する必要がない．

$$S'^T_k (T^m_k S^m_k)^T X_k (T^m_k S^m_k) S'_k \qquad (15)$$

但し，式 (14)では正準形を見つけることができない場合がある．以下の例は，ノードが種類を持たない無向グラフで，Y_4 は X_4 の正準形であり，Z_3 は X_4, Y_4 の第 1 生成行列である．Z_3 は正準形であるので，メモリ上に記憶しておく Z_3 の正準形への変換行列を単位行列とする．

$$X_4 = \begin{array}{c} \\ N_1 \\ N_1 \\ N_1 \\ N_1 \end{array} \begin{array}{c} \begin{array}{cccc} N_1 & N_1 & N_1 & N_1 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$Y_4 = \begin{array}{c} \\ N_1 \\ N_1 \\ N_1 \\ N_1 \end{array} \begin{array}{c} \begin{array}{cccc} N_1 & N_1 & N_1 & N_1 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

$$Z_3 = \begin{array}{c} \\ N_1 \\ N_1 \\ N_1 \end{array} \begin{array}{c} \begin{array}{ccc} N_1 & N_1 & N_1 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

この例において式 (14)を最小にするのは $m = 4$ の時である．X_4 の第 4 ノードを開放除去すると正準形であるので，$S^4_3(X_4), T^4_3(X_4)$ は単位行列となる．$S^4_3(X_4), T^4_3(X_4)$ から $S^4_4(X_4), T^4_4(X_4)$ を計算すると，これらも単位行列となる．従って式 (14)の計算を X_4 に施すと X_4 になる．この計算により X_4 が正しい正準形にならないのは Z_3 の正準形への変換行列が単位行列以外にも

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

などがあるからである．このようにある隣接行列とその正準形の隣接行列の第 1 生成行列が同じである場合に，ある隣接行列の正準形を式 (14)の手法で求めることができない可能性がある．このような場合には X_k のノードの番号の順列により正準形とその変換行列を求める．図 図 6の関数 permutation は X_k のノード番号の順列によりその正準形を求める関数である．但し，X_k の正準形は X_k の第 1 生成行列を第 1 生成行列とする行列であるので効率的に順列をとることができる．

```
1) forall X_k in a set of candidate frequent graphs
2)   X'_k = X_k;
3)   for(m = 1; m ≤ k; m++) do begin
4)     if(N(X_k, k) = N(X_k, m)) then do begin
5)       if(code(X'_k) > code((T_k^m S_k^m)^T X_k (T_k^m S_k^m))
                                    then do begin
6)         X'_k = (T_k^m S_k^m)^T X_k (T_k^m S_k^m);
7)         if(the canonical form of X'_k is known) then do begin
8)           X'_k = S'_k{}^T X'_k S'_k;
               //where S'_k is the matrix to transform
               //X'_k in r.h.s. to its canonical form
9)           break;
10)        end
11)      end
12)    end
13)  end
14)  if(X_k = X'_k)
15)    X'_k=permuation(X_k);
16)  end
17)  Canonical form of X_k is X'_k;
18)  if(X_k ≠ X'_k)
19)    X_k を正準形とする隣接行列があれば X'_k に変更する.
20)  end
21) end
```

図 6 正準化アルゴリズム

§3 頻度計算

全ての多頻度グラフの候補を取り出し正準形を求めた後，実際にデータベースにアクセスする事によって頻度を計算する．ここで数え上げの方法について説明する．図7のようにノードに種類がない有向グラフの場合を考え，大きさ3の多頻度グラフの候補まで計算できているものとする．v_i はトランザクションの第 i ノードである．今は有向グラフについて考えているので隣接行列のコードは4進数で表記されている．さらに大きさ2の隣接行列の中でコードが3以外はグラフは多頻度グラフであるとする．今，大きさが3のグラフの誘導部分グラフの支持度を計算したいので，大きさが1の隣接行列からはじめて，順次サイズの大きな誘導部分グラフの隣接行列を生成していく．v_1, v_2 からなる隣接行列はコードが3で多頻度グラフではないので，この行列はいずれの行列とも組み合わせを行わない．最後に大きさが3の隣接行列のコードが001，111の正準形のカウンタを1つ増やす．1つのトランザクションに対し同型の誘導部分グラフが複数含まれる場合にもカウンタは1つだけ増やす．これは支持度の定義により，支持度はあるグラフが全トランザクションの含まれる割合だからである．ここで数え上げの手法と正規化の違いに注意されたい．正規化の場合は最終的に1つの正規形が分かれば十分であったので結合の組み合わせを限定したが，頻度の計算の場合にはこのような限定はしない．

3·3 実　　装

本稿で提案したアルゴリズムの実装には図8のようにトライデータ構造を用いた．図8はノードが N_1 と N_2 の2つの種類を持つ無向グラフの例である．トライの各接点は正規形の隣接行列を示し，図8の各接点の上段の文字列はノードの種類の系列，下段の数字は隣接行列のコードである．トライの接点の深さはグラフの大きさに相当する．各ノードの親接点はその接点が示すグラフの第1生成行列である．

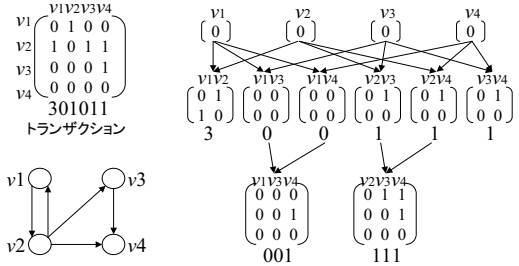

図 7 頻度計算の例

アルゴリズムの実行では root からはじめて，大きさ1のグラフの頻度を計算する．次にそれらを3·2節 3·2·1節で述べた条件1，2，3をもとに結合していく．サイズの1つ大きなグラフを作る場合には，ある隣接行列を第1生成行列として，その行列に対し結合の条件を満たす行列はその行列自身とその右側の接点であるので，このトライデータ構造を用いることによってアルゴリズムを効率的に実行できる．次に結合してできたグラフの各ノードを開放除去したグラフがすべて多頻度グラフであるかどうかをチェックし，すべての多頻度グラフの候補について正準形を求める．さらににデータベースにアクセスし，各トランザクションから誘導部分グラフを全て取り出し，数え上げを行う．最後に各多頻度グラフの候補の支持度が最小支持度を上回ればそれを多頻度グラフとする．以上の操作を繰り返すことにより，ボトムアップに多頻度グラフを順次生成していく．

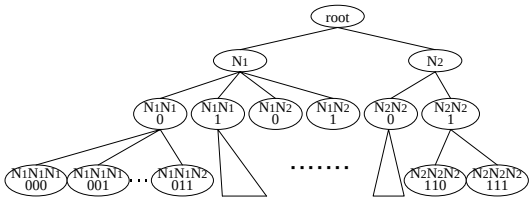

図 8 トライデータ構造による実装

4. 評　価　実　験

本論文で提案するアルゴリズムを実装し，CPU が PentiumII-400MHz 及びメモリが 256MB の PC を用いて評価実験を行った. 表1は本実験で用いたテストデータのパラメータの意味とその基本設定値を要約したものである．はじめに平均 $|T|$，分散1のガウス分布を用いて各トランザクションのサイズを決める．トランザクショ

ン中のノードの種類は等確率で決定する．次にノード間のリンク存在確率 p をもとにノード間にリンクを張る．同様にして，平均サイズ $|I|$ の基本パターンを L 個作る．各トランザクションに対し，基本パターンからランダムに 1 つ選択し上書きする．

図 9, 図 10, 図 11, 図 12はそれぞれトランザクションの数，トランザクションの平均サイズ，ノードの種類の数，最小支持度を変化させて計算時間の評価を行った結果である．変化させたパラメータ以外の値は基本設定値である．図 9を見ると，多頻度グラフの種類はほぼ一定，計算時間はトランザクションの数に比例した．図 10を見ると，計算時間や多頻度グラフの種類は，トランザクションの平均サイズに対し指数関数的に増加する．図 11を見ると，ノードの種類数を増やすと計算時間は指数関数的に減少する．図 12より最小支持度を下げると計算時間や多頻度グラフの種類は増加するが，さらに下げると飽和し，結果として S 字型飽和曲線になると考えられる．従来のバスケット分析で 1 つのトランザクションの同種のアイテムが含まれることはないが，ノードの種類を従来のバスケット分析のデータベース全体に含まれるアイテムの種類と考えると，これらの傾向は従来のバスケット分析と一致する．またノードの種類が 1 の場合に解析できると言うことは，リンクやノードに全く種類がないグラフのトポロジーのみ与えられたデータからの特徴的パターン抽出も可能であることを示している．さらに，有向グラフと無向グラフの結果を比較すると有向グラフの方が計算時間や多頻度グラフの種類は少ない．これは同じノードの数で表現されるグラフの種類は有向グラフの方が非常に多いので，有向グラフの場合，それぞれのグラフに対する頻度は低くなる．そこで有向グラフの場合，グラフの大きさがより小さい段階でも，多くの多頻度グラフの候補の支持度が最小支持度より低くなるために，計算時間は少なくなる．逆に無向グラフの場合は有向グラフの場合に比べて大きいグラフまで最小支持度を下回らず，サイズの大きな多頻度グラフが生成されるので，無向グラフの方が多頻度グラフの種類が多くなる．

図 13はノードの種類が 1 の場合にリンクの存在確率を変化させた場合の結果である．リンクの存在確率が大きくなると，比較的計算時間が少ないことを示している．非正規形は図 8の root の右の枝側に多く存在し，トライの右側は接点の数が比較的少ない．また，トライの右側の接点が示すグラフは比較的リンクの数が多いグラフである．すなわち，非正規形の多くはグラフ構造のリンクの数が比較的多いグラフであり，前処理によりすべのグラフ構造データのリンクの数を多くすることができれば，トライの接点の生成をトライの右側に移行でき，生成される接点の数は減ると考えられる．図 14はノードに種類がない場合の無向グラフの一例で，右側のグラフは左側のグラフの完全グラフに対する補グラフである．完全グラフとは全てのノード間に 1 つのリンクが存在するグラフである．またグラフ G' を G の部分グラフとすると，G に対する G' の補グラフとは G' の全てのリンクを G から取り除いた残りのグラフを言う．図 14のグラフの下の数字はグラフを表す正規形のコードである．図 14の右側のグラフは左側のグラフよりも多くのリンクを持ち，右

表 1 各パラメータの意味

パラメータ	意味	基本設定値
D	トランザクションの数	10,000
$\|T\|$	トランザクションの平均サイズ	10
L	基本パターンの数	10
$\|I\|$	基本パターンの平均サイズ	4
N	ノードの種類	5
p	ノード間のリンク存在確率	50%
$minsup$	最小支持度	10%

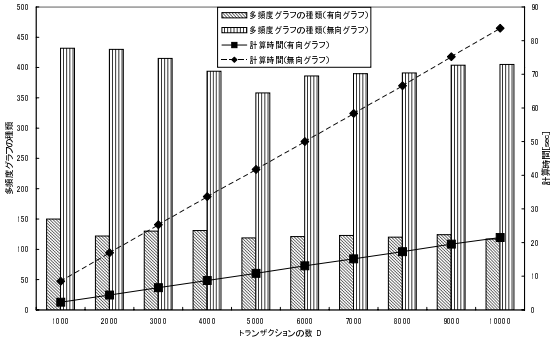

図 9 トランザクションの数の変化に対する計算時間

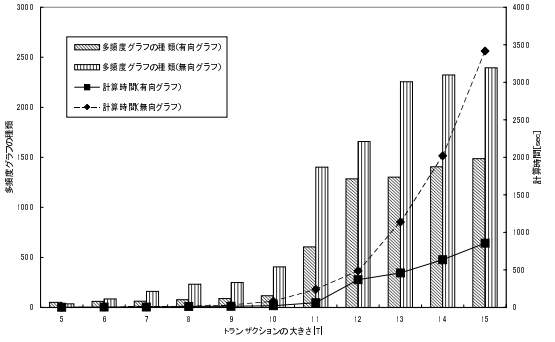

図 10 トランザクションの平均サイズの変化に対する計算時間

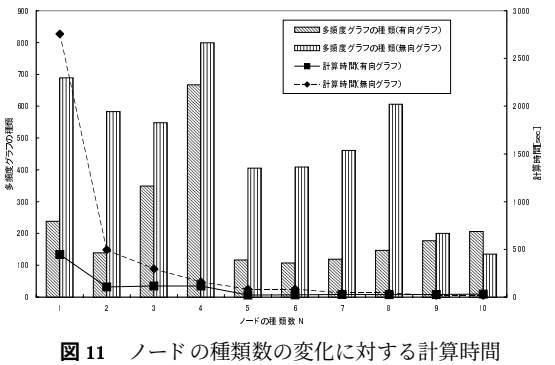

図 11 ノードの種類数の変化に対する計算時間

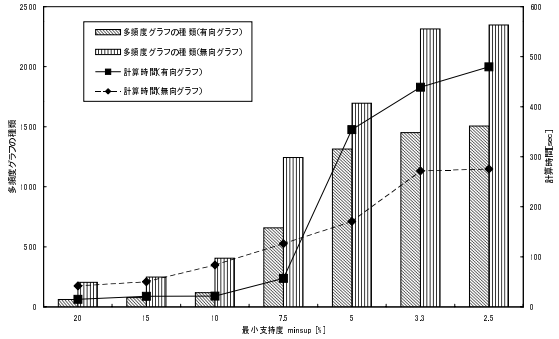

図 12 最小支持度の変化に対する計算時間

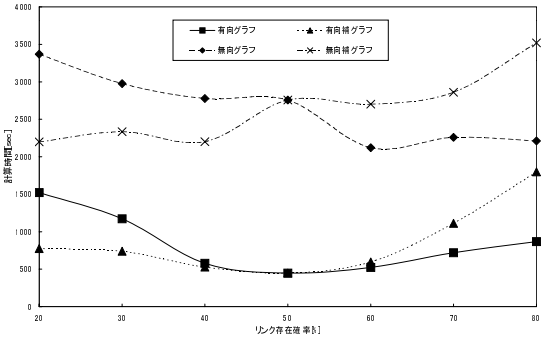

図 13 リンク存在確率の変化に対する計算時間

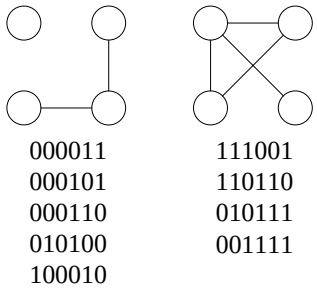

図 14 グラフ構造のコード

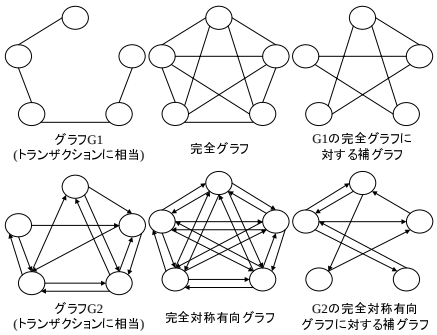

図 15 グラフ構造の補グラフ

側の図のほうが非正規形の隣接行列の数が多い．図 14の左側のグラフを右側のグラフに変換することにより正規形の数を減らすことができる．

そこで無向グラフでトランザクションのノードの数が完全グラフのリンクの数の半分より少ない場合は，図 15のように全てのトランザクションを完全グラフに対する補グラフに変換することによりトライの接点の生成を減らすことができると考えられる．一方，有向グラフの場合には，図 15の下段の中央の図を完全対称有向グラフと定め，このグラフに対する補グラフに変換することによりトライの接点の生成を減らすことができると考えられる．表 2，表 3のように，実際にトランザクションのグラフ構造を補グラフに変換して多頻度グラフを抽出したところ，リンク存在確率が 50%よりも少ない場合は補グラフに変換すると，生成される正規形の数は減り，それにより図 13のように計算時間は減少した．ノードの種類が多いと抽出される多頻度グラフに同種のノードが含まれる可能性が低く，元のデータと補グラフに変換したデータから抽出される正規形の数や計算時間の差がほとんどないと考えられる．そこで補グラフの効果を明確に検証するために，この実験においてはノードの種類を 1 にした．図 11の考察においてリンクの種類が少ない場合に多くの計算時間を要することを述べたが，ノードの種類が少なく，リンク存在確率が 50%以下の場合にはグラフ構造データを補グラフに変換することにより計算時間を減らすことができる．

以上の結果はリンクが種類を持たず自己ループを持たない場合の評価実験の結果であるが，リンクが種類を持つ場合あるいは自己ループを持つ場合も同様の結果が得られた [猪口 2000]．

5. 応　　用

インターネットのサービスの 1 つである WWW では，URL から別の URL へとハイパーリンクが張られ，WWW 利用者はブラウザでハイパーリンクをたどり，次々に URL へアクセスしている．一方，URL を提供する WWW サーバにはクライアントのアクセスに関する情報が貯えられており，そのデータベースのサイズは極めて膨大なものである．そこからユーザの特徴的な経路が得られれば，WWW 管理者やコンテンツの提供者がコンテンツの配置や HTML 文章のレイアウトを考える際に有用であると考えられる．

本研究では（株）リクルートの商用 WWW サイト「あちゃら NAVI」[*1] の WWW サーバへのアクセス履歴の傾向分析を行った．アクセス履歴ファイル[*2] は，ユーザ

*1 現在は MixJuice と統合され，ISIZE に変更．

*2 本実験ではプロキシサーバを考慮していない．厳密な実験をするとなると，クッキーを利用する必要があるだろう．しかし，後にも示すように，厳密ではない実験ではあったが，十分な頻度を持つパターンが得られた．

表 2 リンク存在確率の変化に対する無向グラフの実験結果

リンク存在確率［%］	グラフの状態	多頻度グラフの種類	多頻度グラフの最大サイズ	正規形の種類
20	-	314	9	16,420
	補グラフ			5,751
30	-	438	8	16,759
	補グラフ			8,218
40	-	579	7	16,175
	補グラフ			12,545
50	-	689	7	18,164
60	-	551	7	11,532
	補グラフ			15,390
70	-	414	8	7,870
	補グラフ			19,409
80	-	300	9	5,577
	補グラフ			17,574

表 3 リンク存在確率の変化に対する有向グラフの実験結果

リンク存在確率［%］	グラフの状態	多頻度グラフの種類	多頻度グラフの最大サイズ	正規形の種類
20	-	641	6	11,075
	補グラフ			5,939
30	-	715	5	6,805
	補グラフ			5,102
40	-	368	5	2,924
	補グラフ			2,295
50	-	238	5	1,102
60	-	390	5	2,377
	補グラフ			3,237
70	-	702	6	5,060
	補グラフ			6,797
80	-	674	7	6,472
	補グラフ			12,044

の IP アドレス・アクセス時刻・URL を一行に並べたテキスト形式のファイルである．また，このアクセス履歴ファイルはある 1 日分のデータで，ファイルサイズは約 400MB，約 25,000 種類の IP アドレスが存在した．また，WWW サイト「あちゃら NAVI」において解析の対象とした部分には，約 8,700 個の URL とその間に張られた多数のリンクが存在する．

```
IP アドレス △ アクセス時刻 △ URL
IP1 △ 31/Mar/7:25:45 △ /NAVI/mscategory/Sub/s01/.html
IP1 △ 31/Mar/7:26:37 △ /NAVI/mscategory/Sub/s01/ss06.html
IP2 △ 31/Mar/7:26:38 △ /NAVI/mscategory/Sub/s07.html
IP1 △ 31/Mar/7:27:19 △ /jugemu/home/taidan/member.html
IP2 △ 31/Mar/7:27:23 △ /NAVI/mscategory/Sub/s02.html
                    ⋮
```

△:空白文字

図 16 アクセス履歴レコード仕様

例えば図 17のような 5 つの URL がハイパーリンクで結ばれた WWW サイトを考え，このサイトの利用者は図 17の URL とその間のハイパーリンクを辿っていくとする．ここで，このサイト利用者の多くは A から何らかのパスを通り C に辿り着き，更に E に巡回していくパターンがあるとすると，アクセス履歴ファイルに蓄積された利用者のアクセス経路は，例えば図 18のようなものであると考えられる．ここで C から E に直接のハイパーリンクが張られていないので，このサイトの利用者は C から別の URL に立ち寄って E に至ることになる．図 18の 1 つの系列データは 1 人の利用者のアクセス経路を示している．図 17の矢印がハイパーリンクを表わしたのに対し，図 18ではアクセスの順序を示す．このようなアクセス履歴ファイルから先に述べた A から C を経由して E に至るパターンが得られれば，利用者の便宜のためにハイパーリンクが張られていない C から E に直接のハイパーリンクを張ったほうが良いという知識が得られる．そこで，このような系列データに対し，ある特定の URL ノードのアクセスから n 個先までのアクセスの間に他の特定の URL ノードをアクセスするパターンを調べるために，図 19のように n 個先のノードまでリンクを張る．これを 1 つのトランザクションと考え，多くのトランザクションに含まれる多頻度パターンを抽出する．ここで張るリンクは本稿で提案する手法で A から C を経由して E に至るパターンを抽出するのに必要なリンクである．

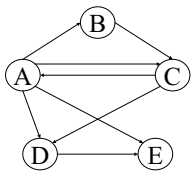

図 17 URL のグラフ

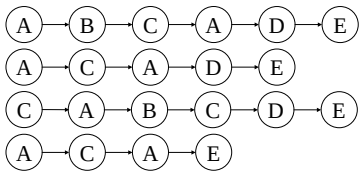

図 18 系列データ

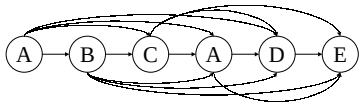

図 19 系列データのグラフ構造への変換（$n = 4$）

先に述べたアクセス履歴ファイルについて，同じ IP アドレスでも 5 分以上アクセスがなかった場合には別のアクセスセッションと見なし，1 回のアクセスセッションを 1 つの系列データとする．さらに，n の値を 5 とし，以上のような前処理を行うと，トランザクション数は 50,666，平均トランザクションサイズは 3.2，最大トランザクションサイズは 118，ノードの種類は 8,655 であった．このデータに対し，最小支持度を 0.15%としてバスケット分析を行うと，1,898 個の多頻度グラフ（最大サイズ 7）が得られ，その計算時間は前節と同じ計算機環境で 52.3 秒であった．

以下に得られたパターンの例を示す．表4は図20，図21，図22のアルファベットとURLの対応表である．図20に示された1つ目の例は6つのURLを順に辿っていった例である．Iから5つ先のURLであるHにリンクが張られているため順にURLを巡回したことが分かる．

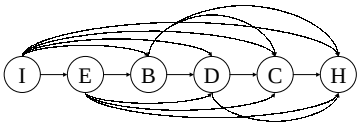

図 20 抽出されたパターン（頻度：76 回）

一方，図21の例は7つのURLを順に巡回した例であるが，Kから5つ先のURLであるBにリンクが張られていないので，Kの直後からBの直前までに1回別のURLに立ち寄った可能性があることを示している．

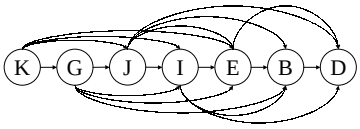

図 21 抽出されたパターン（頻度：80 回）

図22は我々の手法が互いに連続せず分離した特徴的パターンを抽出可能であることを示している．

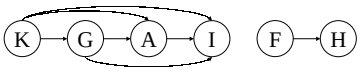

図 22 抽出されたパターン（頻度：77 回）

このような多頻度アクセスパターンがどのように役に立つのか考える．例えば，図20，図21のパターンはそれぞれのURLに対し異なったバナー広告を配置することで，このサイトを訪れたアクセス者に多くの広告を見せるというマーケティング戦略に使える．また図21はKからBの直前までに1回別のURLに立ち寄った可能性があるので，7つのURLの間のどれかはハイパーリンクが張られていない可能性がある．図22のパターンもURLが互いにハイパーリンクが張られてい可能性があるので，これらのパターンはハイパーリンクを張るための候補をあげる知識となりうる．

6. 関 連 研 究

グラフ構造データからの特徴的パターンの抽出は機械学習の分野でも研究されており，その代表的な手法にGBI（Graph Based Induction）がある [Motoda 97]．GBIでは多くの構造データは有向グラフで表現できることに着目し，グラフの中から「特徴的ペアの逐次拡張」という単純な操作を繰り返し適用することによって，グラフ中に頻発に現れる特徴的なパターンを効率良く抽出する方法である．GBIの利点は頻度以外にも情報エントロピーやgini-indexなどの指標を用いることができる点である．しかし，GBIはGreedy探索を採用しているため，実世界の大量データにおいてすべてのパターン探索には不向きである．またGBIでは連続せず分離した位置にある特徴的パターン抽出が不可能である．

表 4 対応表

記号	URL	ページタイトル
A	s01	アート
B	s03	ニュース・情報
C	s04	インターネット
D	s05	スポーツ
E	s06	エンターテイメント
F	s08	マルチメディア
G	s09	飲食
H	s12	健康・医療・美容
I	s13	社会・文化
J	s14	趣味・生活
K	s16	旅行

s01は/NAVI/mscategory/Sub/s01.html の意.

一方，与えられた2つのグラフの同型性判定問題についてはNAUTY[McKay 90] と呼ばれる有名なシステムがある．これはグラフの不変量を利用して与えられたグラフを部分構造に分解し，効率的にノードのマッチングを行うことにより同型性を判定する．NAUTYは2つのグラフの同型性判定を行うので大量のグラフ構造データを扱うことは不可能である．またNAUTYではグラフ中の各ノードの線度などの不変量を利用して同型性判定を行うが，あるグラフ構造データから誘導部分グラフを取り出すと各ノードに対する線度は変わるので，この誘導部分グラフに対し不変量を計算し直す必要があり，有限の計算時間内で部分グラフの共起性を調べることは不可能であると考えられる．

7. お わ り に

本稿ではAprioriアルゴリズムを拡張することにより，グラフ構造データから多頻度パターンを効率的に抽出するアルゴリズムの提案した．テストデータを用いた評価実験では従来のバスケット分析と同様の結果が得られた．今後の課題としてはより複雑な構造を持つデータの分析とさらなる効率化を検討中である．

謝　　辞

本研究の一部は，文部省科学研究費基盤研究(B)(2)(12480088)の支援によって行われた．

◇ 参 考 文 献 ◇

[Agrawal94] Agrawal, R and Srikant, R.: Fast Algorithm for Mining Association Rules in Large Databases. *Proc. of the 20th Very Large Data Bases Conference*, pp. 487-499, 1994.

[Arimura 98] Arimura, H., Wataki, A., Fujino, R. and

Arikawa, S.: A Fast Algorithm for Discovering Optimal String Pattern in Large Text Database. *Proc. of the 9th International Conference on Algorithmic Learning Theory*, pp. 247-261, 1998.

[猪口 2000] 猪口 明博：膨大なグラフ構造データからの高速マイニング手法に関する研究．大阪大学大学院工学研究科修士論文, 2000.

[前原 99] 前原 恵太, 上原 邦昭：並列組織化アルゴリズムによるグラフマッチングと部分構造発見手法．電子情報通信学会論文誌 D-I Vol. J82-D-I No. 1 pp. 1-10, 1999.

[McKay 90] McKay, B.D.: NAUTY User's Guide (version 1.5). *Technical Report, TR-CS-90-02*, Department of Computer Science, Australian National University, 1990.

[Motoda 97] Motoda, M. and Yoshida, K.: Machine Learning Techniques to Make Computers Easier to Use. *Proc. of the 15th International Joint Conference on Artificial Intelligence*, Vol. 2, pp. 1622-1631, 1997.

[元田 99] 元田 浩：明示的理解に魅せられて, 人工知能学会誌, Vol. 14, No. 4, pp. 615-625, 1999.

[Shintani 98] Shintani, T. and Kituregawa, M: Mining Algorithms for Sequential Patterns in Parallel: Hash Based Approach. *Proc. of the 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 283-294, 1998.

[Yoshida 95] Yoshida, K. and Motoda, H.: CLIP: Concept Learning from Inference Patterns. *Artificial Intelligence*, Vol. 75, No. 1 pp. 63-92, 1995.

〔担当委員：××○○〕

19YY 年 MM 月 DD 日　受理

著　者　紹　介

猪口　明博(正会員)

inokuchi@jp.ibm.com

1998 年大阪大学工学部通信工学科卒業．2000 年同大学院工学研究科通信専攻博士前期課程修了．同年，日本アイ・ビー・エム（株）に入社．現在，同社東京基礎研究所に所属．データマイニングに関する研究に興味を持つ．

鷲尾　隆(正会員)

washio@sanken.osaka-u.ac.jp

1960 年生．1983 年東北大学工学部原子核工学科卒業．1988 年東北大学大学院原子核工学専攻博士課程修了．工学博士．1988 年から 1990 年にかけてマセチューセッツ工科大学原子炉研究所客員研究員．1990 年（株）三菱総合研究所入社．1996 年退社．現在，大阪大学産業科学研究所助教授（知能システム科学研究部門）原子力システムの異常診断手法に関する研究，定性推論に関する研究を経て，現在は人工知能の基礎研究，特に科学的知識発見，データマイニングなどの研究に従事．著書に "*Expert Systems Applications within the Nuclear Industry*", American Nuclear Society,「知能工学概論」: 第 2 章エージェント（共著，廣田　薫　編，昭晃堂）など．人工知能学会，計測自動制御学会，日本ファジイ学会，情報処理学会，AAAI, 各会員．

元田　浩(正会員)

motoda@sanken.osaka-u.ac.jp

1943 年生．1965 年東京大学工学部原子力工学科卒業．1967 年東京大学大学院原子力工学専攻修士課程修了．同年（株）日立製作所入社．同社中央研究所，原子力研究所，エネルギー研究所，基礎研究所を経て 1995 年退社．現在，大阪大学産業科学研究所教授（知能システム科学研究部門）原子力システムの設計，運用，制御に関する研究，診断型エキスパート・システムの研究を経，現在は人工知能の基礎研究，特に機械学習，知識獲得，知識発見などの研究に従事．工学博士．認知科学会，人工知能学会，情報処理学会，日本ソフトウエア科学会，AAAI, IEEE Computer Society，各会員．

熊澤　公平

kumazawa@isr.recruit.co.jp

1962 年生．1987 年東北大学工学部機械工学専攻修了．1987 年株式会社リクルート入社．1987 年から 1995 年までスーパーコンピュータ SI の事業に従事．1995 年から同社 WWW サービスのシステムを構築．現在,（株）リクルート次世代開発グループメディアデザインセンターにて，分散オブジェクト，ソフトウエアエージェントなどの実証実験に従事．

荒井　尚英

arai@in.recruit.co.jp

1963 年生．1986 年東北大学工学部精密工学科卒業．1986 年（株）リクルート入社 1992 年早稲田大学システム科学研究所ビジネススクール卒業．1997 年から 1999 年にかけて Arthur Andersen L.L.P. ビジネスコンサルティング部門ロスアンゼルスオフィスにて，コンサルタントとして従事．現在,（株）リクルート次世代事業開発室にてインターネット上での金融関連事業開発に従事．論文・翻訳に「AMEX vs.VISA」ダイヤモンド・ハーバードビジネス誌（募集論文）1992 年 11 月,「ハイエンド市場を支配し続けるカオス創出戦略（シリコングラフィクス社 CEO　エドワード・マクラッケン氏インタビュー）」ダイヤモンド・ハーバードビジネス誌（翻訳）1994 年 3 月など．日本グラフィックデザイナー協会会員．