Fast Apriori-based Graph Mining Algorithm (AGM) とその性能評価

西村 芳男*† 鷲尾 隆† 吉田 哲也† 元田 浩† 猪口 明博‡

†大阪大学 産業科学研究所 ‡日本アイ・ビー・エム (株) 東京基礎研究所

† 〒 567-0047 大阪府茨木市美穂ヶ丘 8-1
Tel:(06)6879-8542, Fax:(06)6879-8544
E-Mail: nishimura@ar.sanken.osaka-u.ac.jp

Apriori-based Graph Mining (AGM) アルゴリズムは，グラフ構造データベースから多頻度の誘導部分グラフ構造パターンを抽出するように，Apriori アルゴリズムを拡張したものである．AGM アルゴリズムは多くの高速化アルゴリズムが提案されており，本稿では多頻度グラフの候補生成を高速化する AGM' アルゴリズム, 支持度計算を高速化する AGM* アルゴリズム，AGM' アルゴリズムと AGM* アルゴリズムを併用した AGM'* アルゴリズムについて，人工データと実データによる性能評価と特徴分析を行った．

1. まえがき

グラフ構造データベースの中に部分構造として含まれる特徴的なパターンを完全探索によって抽出する AGM アルゴリズムが猪口によって提案された [Inokuchi 00]．AGM アルゴリズムは，Apriori アルゴリズム [Agrawal 94] をグラフ構造データに拡張したアルゴリズムであり，その抽出方法の効率化が行われた [猪口 01] [猪口 02] [西村 02]．本稿では，多頻度グラフの候補生成を高速化する AGM' アルゴリズム [西村 02] , 支持度計算を高速化する AGM* アルゴリズム [猪口 02] と AGM' アルゴリズムと AGM* アルゴリズムの両方を併用した AGM'* アルゴリズムについて，人工データと実データによる性能評価と特徴分析を行う．

2. AGM アルゴリズム

AGM アルゴリズム [Inokuchi 00] は，グラフ構造データベース GD が与えられたとき，ユーザが指定した最小支持度 (minsup) と呼ばれる閾値を使用して，GD の中に最小支持度を上回る支持度で誘導部分グラフとして含まれるグラフ構造のみを効率よく抽出するアルゴリズムである．グラフ G_s の支持度 $sup(G_s)$ は，

$$sup(G_s) = \frac{G_s\text{を誘導部分グラフとして含むグラフの数}}{GD\text{に含まれるグラフの総数}}$$

で定義され，グラフ G_s の支持度 $sup(G_s)$ が最小支持度を上回る場合，グラフ G_s を多頻度グラフと呼ぶ．

AGM アルゴリズムは，頂点数が 1 の多頻度グラフをから逐次的に頂点数が多い多頻度グラフをレベルワイズに抽出する．図 1 に AGM アルゴリズムの概略を示す．始めに，頂点数が 1 の多頻度グラフをデータベースより抽出し，それを F_1 に代入する．次に，関数 apriori-gen-join では，頂点数が k の多頻度グラフから頂点数 $k+1$ の多頻度グラフの候補を生成し，それを $\hat{C}_{k+1}$ に代入する．次に，関数 apriori-gen-prune では $\hat{C}_{k+1}$ に格納されている多頻度グラフの各候補について，多頻度グラフであるための必要条件を調べる．この条件を調べることで多頻度グラフの候補の数を絞りこむ．絞りこみで残った多頻度グラフの候補のみを C_{k+1} に格納する．次に，関数 count ではグラフ構造データベース GD にアクセスして，C_{k+1} の各

```
// GD:グラフ構造データベース
// F_k:頂点数 k の多頻度グラフの集合
// Ĉ_{k+1}:頂点数 k の多頻度グラフを合成したものの集合
// C_k:頂点数 k の多頻度グラフの候補の集合
// minsup:最小支持度 (閾値)
1) F_1 = { Frequent subgraph of size=1} ;
2) for(k = 1; F_k ≠ ∅; k++) do begin
3)    Ĉ_{k+1} =apriori-gen-join(F_k);
4)    C_{k+1} =apriori-gen-prune(Ĉ_{k+1});
5)    count(GD, C_{k+1});
6)    F_{k+1} ={ c_{k+1} ∈ C_{k+1} | sup(G(c_{k+1})) ≥ minsup};
7) end
8) Answer= ⋃_k F_k ;
```

図 1: AGM アルゴリズム

要素の支持度を求める．C_{k+1} の各要素の支持度が最小支持度を上回る場合は，そのグラフを多頻度グラフとし，それを F_{k+1} に格納する．以上の操作を F_k が空集合になるまで繰り返し，グラフ構造データベース GD に含まれる多頻度グラフをすべて抽出する．

2.1 定義

AGM アルゴリズム で扱うグラフは頂点，辺にラベルを持ち，以下のように定義される．

定義 1 (ラベル付きグラフ) 頂点の集合 $V(G)$，辺の集合 $E(G)$，頂点のラベル集合 $L_V(V(G))$，辺のラベル集合 $L_E(E(G))$ が

$$\begin{aligned} V(G) &= \{\ v_1, v_2, ..., v_k\ \}\ , \\ E(G) &= \{\ e_h = (v_i, v_j) | v_i, v_j \in V(G), i \neq j\}\ , \\ L_V(V(G)) &= \{lb(v_i) | \forall v_i \in V(G)\ \}\ , \\ L_E(E(G)) &= \{lb(e_h) | \forall e_h \in E(G)\ \} \end{aligned}$$

と与えられたとき，グラフ G は

$$G = (V(G), E(G), L_V(V(G)), L_E(E(G)))$$

と表現される．ここで，頂点の数 $|V(G)| = k$ をグラフ G の大きさとする．$lb(v_i)$ および $lb(e_h)$ はそれぞれ 頂点 v_i のラベル，辺 e_h のラベルである．頂点ラベル $lb(v_i)$ および辺ラベル $lb(e_h)$ にはそれぞれ $num(lb(v_i))$,$num(lb(e_h))$ によって自然数を以下のように割り当てる．

定義 2 (ラベル間の順序関係) グラフ構造データベースが与えられたとき，それに含まれる各ラベル lb_i の頂点の数を $avg(lb_i)$ とする．$avg(lb_i)$ が少ないものから自然数を昇順に割り当てると，生成されるグラフ数は少なくなる *[猪口 01]*．つまり，

$$if\ avg(lb_i) < avg(lb_j)\ then\ num(lb_i) < num(lb_j)\quad for\ i, j = 1, ..., |L_V(V(G))|, i \neq j$$

とする．辺のラベル lb_i に割り当てる自然数も同様に，

$$if\ avg(lb_i) < avg(lb_j)\ then\ num(lb_i) < num(lb_j)\quad for\ i, j = 1, ..., |L_E(E(G))|, i \neq j$$

とする．

定義 3 (隣接行列) 大きさ k のグラフ $G = (V(G), E(G), L_V(V(G)), L_E(E(G)))$ が与えられたとき，隣接行列 X_k の (i, j) 要素 $x_{i,j}$ は

$$x_{i,j} = \begin{cases} num(lb(e_h)) & if\ e_h = (v_i, v_j) \in E(G) \\ 0 & if\ (v_i, v_j) \notin E(G) \end{cases}$$

で与えられる．ただし，グラフ G は頂点ラベルに割り当てられた自然数によって，

$$num(lb(v_i)) \leq num(lb(v_{i+1}))\ for\ i = 1, ..., k-1$$

の条件を満たすように頂点を並び替える．頂点の並び替えを行うと隣接行列は異なった行列となり，それらの隣接行列は以下の変換行列によってお互いに変換することができる．

定義 4 (変換行列) 頂点数が k の隣接行列 X_k, Y_k が同型なグラフ構造を表すとき，X_k から Y_k への変換行列 W_k の要素 w_{ij} は，

$$w_{ij} = \begin{cases} 1 & G(X_k) \text{ の第 } i \text{ 頂点が } G(Y_k) \text{ の第 } j \text{ 頂点 に相当する時} \\ 0 & \text{その他} \end{cases}$$

と定義され, Y_k は $Y_k = W_k^T X_k W_k$ となる．

2.2 章で述べる多頻度グラフの候補の生成数を減らすために，グラフのコード を以下のように定義する．

定義 5 (グラフのコード) 隣接行列のコード $code(X_k)$ は隣接行列 X_k の要素 $x_{i,j}$ を並べたものであり，有向グラフの場合には，

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

と定義する．無向グラフの場合は $x_{i,j} = x_{j,i}$ であるため冗長なものを省いて，

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

と定義する．さらに頂点ラベルを含めたグラフのコード $CODE(G(X_k))$ を

$$CODE(G(X_k)) = num(lb(v_1))\cdots num(lb(v_k))code(X_k)$$

と定義する．

定義 6 (正準形) グラフ $G(X_k)$ と同じ構造をもつグラフの集合を $\Gamma(G(X_k))$ としたとき，$\Gamma(G(X_k))$ の中で最小の $CODE$ を持つグラフ g を正準形 *(canonical form)* とする．

$$g \quad w.r.t \quad CODE(g) = \min_{\forall g_i \in \Gamma(G(X_k))} CODE(g_i)$$

2.2 多頻度グラフの候補生成 (join 部)

多頻度グラフの候補生成は，大きさが k の多頻度グラフを合成して大きさ $k+1$ の多頻度グラフの候補を作成する join 部と，合成された多頻度グラフの候補が多頻度グラフになるための必要条件を満たすかどうかを調べる prune 部の 2 つの部分から成り立つ．join 部では以下の条件を満たすように多頻度グラフの候補 $G(Z_{k+1})$ を順に生成していく．

条件 1 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，すなわち各グラフの第 k 頂点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，大きさ $k+1$ の 隣接行列 Z_{k+1} を生成する．

$$X_k = \begin{pmatrix} X_{k-1} & \boldsymbol{x}_1 \\ \boldsymbol{x}_2^T & 0 \end{pmatrix}, Y_k = \begin{pmatrix} X_{k-1} & \boldsymbol{y}_1 \\ \boldsymbol{y}_2^T & 0 \end{pmatrix} \Longrightarrow Z_{k+1} = \begin{pmatrix} X_{k-1} & \boldsymbol{x}_1 & \boldsymbol{y}_1 \\ \boldsymbol{x}_2^T & 0 & z_{k,k+1} \\ \boldsymbol{y}_2^T & z_{k+1,k} & 0 \end{pmatrix}$$

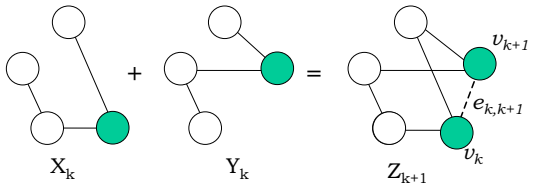


図 2: 多頻度グラフの候補生成例

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．$G(X_k)$，$G(Y_k)$ をそれぞれ $G(Z_{k+1})$ の第 *1* 生成グラフ，第 *2* 生成グラフと呼ぶ．

条件 2 $i = 1, \cdots, k-1$ として，生成される $G(Z_{k+1})$ の頂点ラベルには以下の条件がある．

$$\begin{aligned} lb(v_i \in V(G(Z_{k+1}))) &= lb(v_i \in V(G(X_k))) \\ &= lb(v_i \in V(G(Y_k))), \\ num(lb(v_i \in V(G(X_k)))) &\leq num(lb(v_{i+1} \in V(G(X_k)))), \\ lb(v_k \in V(G(Z_{k+1}))) &= lb(v_k \in V(G(X_k))), \\ lb(v_{k+1} \in V(G(Z_{k+1}))) &= lb(v_k \in V(G(Y_k))), \\ num(lb(v_k \in V(G(X_k)))) &\leq num(lb(v_k \in V(G(Y_k)))) \end{aligned}$$

ただし，隣接行列 Z_{k+1} の $(k, k+1)$ 要素 $z_{k,k+1}$ および $(k+1, k)$ 要素 $z_{k+1,k}$ は X_k, Y_k から決定することはできない．

条件 3 そこで，隣接行列 Z_{k+1} は以下の条件を満たすものすべてが作られる．すなわち，

$$\hat{C}_{k+1} \leftarrow G(Z_{k+1}) \ where \ z_{k,k+1} = lb1 \ and \ z_{k+1,k} = lb2,$$
$$\forall \ lb1, 0 \leq lb1 \leq |L_E(E(G))| \ and \ \forall \ lb2, 0 \leq lb2 \leq |L_E(E(G))|$$

である．有向グラフの場合は $(|L_E(E(G))| + 1)^2$ 個のグラフが生成される．無向グラフの場合は $z_{k,k+1} = z_{k+1,k}$ であるため，$(|L_E(E(G))| + 1)$ 個のグラフが生成される．

条件 4 ここでグラフ構造 $G(X_k)$ と $G(Y_k)$ の第 k 頂点のラベルが等しい場合，$G(Y_k), G(X_k)$ をそれぞれ第 *1* 生成グラフ，第 *2* 生成グラフとして *2* つのグラフを結合した場合，このグラフは冗長である．そこで，このような冗長な生成を避けるため，以下の関係にある場合にのみグラフを結合する．

$$CODE(\text{第 } 1 \text{ 生成グラフ}) \leq CODE(\text{第 } 2 \text{ 生成グラフ})$$

以上の *4* つの条件のもとで生成されるグラフを**正規形**（$normal\ form$）と呼ぶ．

2.3 多頻度グラフの候補生成 (prune 部)

前節の join 部で合成された多頻度グラフの候補 $G(Z_{k+1}) \in \hat{C}_{k+1}$ が多頻度グラフであるための必要条件は，$G(Z_{k+1})$ の全ての誘導部分グラフが多頻度グラフであることである．そこで，この必要条件と等価である以下の必要条件を調べる．

誘導部分グラフの必要条件
グラフ $G(Z_{k+1})$ が多頻度グラフであるための必要条件は，$G(Z_{k+1})$ の第 $i(1 \leq i \leq k-1)$ を除去してできるグラフが全て多頻度グラフであることである．

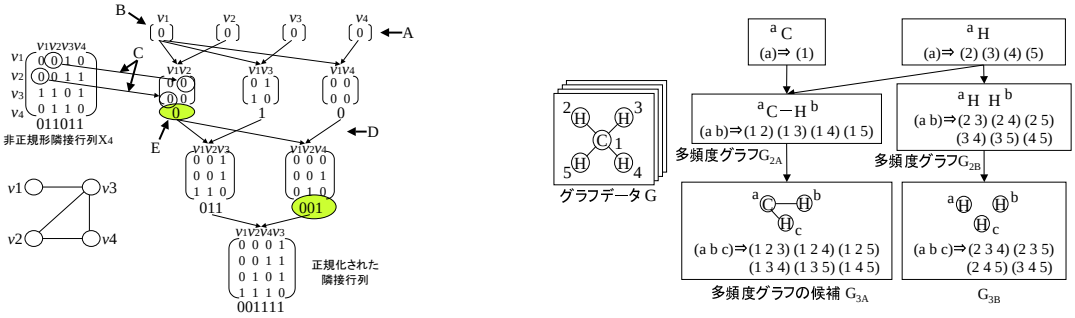
先にも述べたように，このアルゴリズムでは正規形の隣接行列しか探索生成しないために，第 i 頂点を開放除去したグラフの隣接行列が正規形でなければ，それが多頻度グラフであるかを過去の探索から容易にチェックする事ができない．よって，非正規形の隣接行列を正規化する手法が必要である．

正規化の具体例を図 3(a) の非正規形の隣接行列 X_4 の正規化で示す．(A) はじめに頂点が 1 つからなる 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．(D) 次に頂点数が 2 の隣接行列の結合を行う．(E) このとき隣接行列のコードが最小の行列を第 1 生成行列とする．ここではコードが 0 の隣接行列が 2 つあるが，どちらか一方を選択する．以下，順に繰り返し，非正規形の隣接行列 X_4 を再構築し正規化された行列を得る．上記の方法によって頂点を除去した全ての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り，C_{k+1} に格納される．

全ての多頻度グラフの候補を取り出した後，実際にデータベースをスキャンして，それらの支持度計算する．しかし，正規形の中にも同じグラフ構造を表すグラフが存在する場合があるため，支持度を計算する前に正準形を求める処理が必要となる．正準形を求める処理は文献 [Inokuchi 00] を参照されたい．

2.4 支持度計算

支持度計算はグラフデータ G に含まれる誘導部分グラフを頂点数が 1 のものから，順次頂点数が大きなものをレベルワイズにすべて抽出する．今，図 3(b) のように頂点のラベルが “C”,“H” の 2 種類, 辺のラベルが “—” の 1 種類からなるグラフで，頂点数が 3 の多頻度グラフの候補の支持度を計算する場合を考える．グラフデータ G の頂点に割り当てられた数字と多頻度グラフの候補に割り当てられたアルファベットは頂点の ID とする．抽出する誘導部分グラフはグラフデータ G の頂点 ID の順列で表現される．例えば，グラフデータ G の誘導部分グラフで頂点数が 2 のものは，(1 2),(1 3),(1 4),(1 5),(2 3),(2 4),(2 5),(3 4),(3 5),(4 5) の 10 個があり，この中で多頻度グラフの構造を表すものだけを抽出する．この例の場合はグラフ G_{2A} と G_{2B} の両方が多頻度グラフなので，すべての頂点 ID の順列を抽出している．同様にして，頂点数が 3 の誘導部分グラフを表す頂点 ID の順列を抽出する．最後に頂点数が 3 のグラフ G_{3A}，G_{3B} に対応する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形の誘導部分グラフが複数含まれる場合に



(a) 正規化の例　　(b) 支持度計算の例

図 3: 正規化と支持度計算の例

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．

このようにして，C_{k+1} の各要素について多頻度グラフの候補の支持度を計算し，その支持度が最小支持度を上回る場合には，その多頻度グラフの候補を多頻度グラフとして，F_{k+1} に格納する．つまり，

$$\text{if } \forall c_{k+1} \in C_{k+1} \text{ and } sup(G(c_{k+1})) \geq minsup \text{ then } F_{k+1} \leftarrow c_{k+1}$$

である．

3. AGM の高速化アルゴリズム

AGM アルゴリズムの高速化が多く提案されている．AGM'アルゴリズム [西村 02] は，2.2 章で説明した多頻度グラフの候補生成を高速化する．2.4 章で説明した支持度計算を高速化するアルゴリズム [猪口 02] を AGM* とし，この 2 つのアルゴリズムの説明を行う．

3.1 AGM' アルゴリズム

2.2 章の図 2 のように，大きさが k の多頻度グラフ $G(X_k)$ と $G(Y_k)$ を結合し，大きさが $k+1$ の多頻度グラフの候補 $G(Z_{k+1})$ を生成する場合を考える．グラフ $G(Z_{k+1})$ の要素 $z_{k,k+1}$ と $z_{k+1,k}$ は X_k, Y_k から決定することができないため，辺のラベル数 $|L_E(E(G))|$ に応じて条件 3 で示された数のグラフ $G(Z_{k+1})$ が生成される．生成されたグラフ $G(Z_{k+1})$ に含まれるすべての誘導部分グラフが多頻度グラフであることを確認するために，それと等価な必要条件を確認している．この方法では合成されたすべてのグラフ $G(Z_{k+1})$ について，その誘導部分グラフを正規化する必要があるため多くの計算時間を要する．

そこで，$G(Z_{k+1})$ の頂点 v_k, v_{k+1} とその頂点間の辺 $e_{k,k+1} = (v_k, v_{k+1})$ から構成される大きさ 2 のグラフ $G(Z_2)$ に着目する．$G(Z_{k+1})$ が多頻度グラフになるためには，その誘導部分グラフである $G(Z_2)$ も多頻度グラフであることが必要条件の 1 つである．つまり，$G(Z_2)$ が多頻度グラフでない場合は，$G(Z_{k+1})$ も多頻度グラフになり得ない．そのため，このような合成を行わない．そこで，$k \geq 2$ では条件 3 の代わりに以下の条件 3' を使用する．

条件 3'

$$\text{If } k \geq 2 \text{ and } \forall\, G(Z_2) \in F_2 \text{ then } \hat{C}_{k+1} \leftarrow G(Z_{k+1}), \text{ where } V(G(Z_2)) = \{v_k, v_{k+1}\}$$

条件 1, 条件 2, 条件 4 と条件 3' を満たすグラフ $G(Z_{k+1})$ を改めて正規形と定義する．AGM'アルゴリズムで合成された $G(Z_{k+1}) \in \hat{C}_{k+1}$ は 2.3 章で述べた必要条件を確認するために $G(Z_{k+1})$ の誘導部分グラフを正規化する必要がある．しかし，従来の AGM アルゴリズムと比較して，AGM'アルゴリズムの $\hat{C}_{k+1}$ は要素が絞り込まれているため，正規化の計算時間を短縮できる．

3.2 AGM*の支持度計算

AGM アルゴリズムの支持度計算では，グラフデータ G に含まれる頂点数 k の誘導部分グラフ G_k をすべて抽出する．しかし，支持度の定義はグラフ G_k を誘導部分グラフとして含むグラフデータ G の割合であるから，グラフ G_k がグラフデータ G に含まれているかがわかればよい．そこで, AGM*アルゴリズム [猪口 02] ではグラフデータ G と多頻度グラフの各候補 G_k 間の写像を 1 対 1 で調べる．探索方法は深さ優先探索を行い，グラフ G_k がグラフデータ G に含まれていることが発見できれば，そこで探索を終了する．さらに，以下の方法で探索する空間の効率化を行っている．

多頻度グラフの候補 G_k の第一生成グラフを G_{k-1} とする．図 3(b) の頂点に割り当てられた数字およびアルファベットは頂点の ID とする．G_k が G に含まれるか調べる時，過去の支持度計算によって $G_{k-1} \Rightarrow G$ の写像を示す探索木のパス (a,b)⇒(1,5) が見つかっており，このパスより左側は探索が終了している．したがって，$G_k \Rightarrow G$ の写像を調べる際は，このパス (a,b)⇒(1,5) より探索を開始すればよい．探索の結果，$G_k \Rightarrow G$ の写像が (a,b,c)⇒(1,5,6) のパスであるとわかる．この結果は次回の探索時に利用するため，保存しておく．このように，AGM*では第一生成グラフ G_{k-1} の探索結果を利用し，探索木の一部だけを探索するように効率化されている．

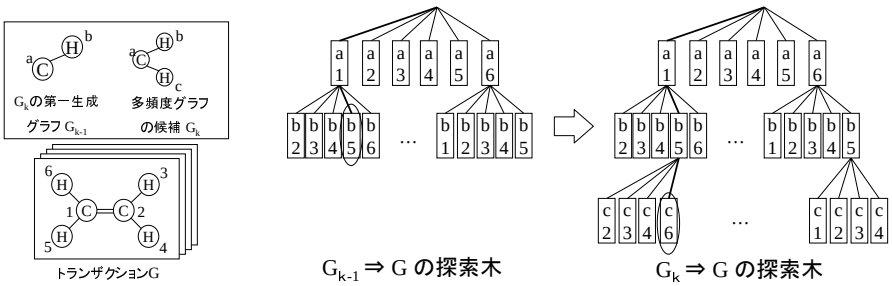


図 4: AGM*の支持度計算

表 1: 人工データのパラメータとそのデフォルト値

パラメータ	意味	デフォルト値
D	GD に含まれるグラフ数	1000
$\|T\|$	グラフデータの平均頂点数	15
L	基本パターンの数	8
$\|I\|$	基本パターンの平均頂点数	7
$\|L_V\|$	頂点ラベルの種類数	5
$\|L_E\|$	辺ラベル種類数	5
p	頂点間に辺が存在する確率	4%
$minsup$	最小支持度	10%

4. 評価実験

本研究の提案手法を C 言語で実装し，CPU が PentiumIII 1GHz, メモリが 1.5GB 搭載された計算機を使用して，評価実験を行った．表 1 に実験で用いた人工データのパラメータとそのデフォルト値を示す．まず，平均 $|T|$，分散 1 のガウス分布を用いて各グラフ構造データのサイズを決める．各グラフ構造データ中の頂点のラベルは等確率で決定する．次に存在確率 p をもとに頂点間に辺を結ぶ．辺のラベルも等確率で決定する．同様にして，平均サイズ $|I|$ の基本パターンを L 個作る．各グラフ構造データに対し，基本パターンからランダムに 1 つを選択し，上記グラフ構造データが基本パターンを誘導部分グラフとして含むように上書きする．

4.1 AGM と AGM' の比較結果

AGM と AGM' の多頻度グラフの候補生成の計算時間を比較した結果を図 5 に示す．図 5(a) は辺のラベル数 $|L_E|$, 図 5(b) は頂点のラベル数 $|L_V|$, 図 5(c) はグラフデータの平均頂点数 $|T|$ を変化させた場合の，多頻度グラフの候補生成に要する計算時間を評価した結果である．辺のラベル数 $|L_E|$ および頂点のラベル数 $|L_V|$ が少ない場合に，AGM' は多頻度グラフの候補生成における計算時間をほとんど削減できていない．これはラベル数が少なく，グラフ構造データベースに多くの多頻度グラフが多く含まれているためである．しかし，辺のラベル数 $|L_E|$ および頂点のラベル数 $|L_V|$ が多い場合には，AGM' は多頻度グラフの候補生成において，計算時間を削減できている．また，$|L_E| \geq 3$ および $|L_V| \geq 7$ のようにある閾値を超えると，抽出される多頻度グラフは基本パターンに含まれるものがほとんどであるため，AGM' の計算時間はほぼ一定になると考えられる．

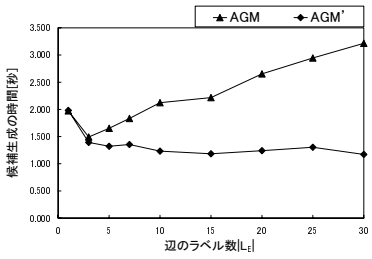
また，図 5(c) のようにグラフデータの平均頂点数 $|T|$ を増やした場合にも，AGM' の条件 3' は多頻度グラフの候補生成にかかる計算時間を削減できている．平均頂点数がより大きなグラフデータに対しても，AGM' が計算時間を削減することが期待できる．

図 5(d) は各頂点間に存在する辺の割合を $p = 1\%$ に, 図 5(e) は $p = 30\%$ に固定し，最小支持度 $minsup$ を変化させた場合の多頻度グラフの候補生成に要する計算時間を評価した結果である．この 2 つの場合を比較すると，AGM' が有効な場合は頂点間の辺の存在する確率 p が低いグラフ (疎グラフ) でかつ，最小支持度 $minsup$ が低い時である．

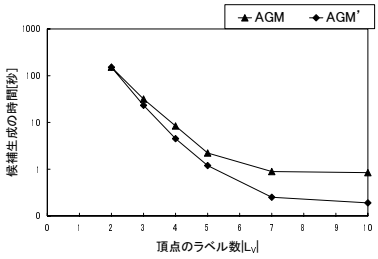
4.2 AGM と AGM*の比較結果

図 6 は AGM，AGM'，AGM*の各アルゴリズムをシステム全体の計算時間で比較した結果である．図 6(a) は各頂点間に存在する辺の割合を $p = 1\%$ に, 図 6(b) は $p = 30\%$ に固定し，最小支持度 $minsup$ を変化させた場合の計算時間の結果である．AGM と AGM*アルゴリズムを比較した場合，AGM の支持度計算はグラフデータに含まれる誘導部分グラフ全てを一度の操作で抽出できるため，$p = 30\%$ では AGM のほうが高速である．しかし，AGM の支持度計算はグラフデータに含まれる頂点 ID の順列を全て取り出すため，多頻度グラフ数が増加するにつれて計算時間の増加も大きく，疎グラフデータである $p = 1\%$ では AGM*のほうが高速である．図 6(c) は基本パターンの平均頂点数 $|I|$，図 6(d) はグラフデータの平均頂点数 $|T|$ を変化させた場合の計算時間の結果である．AGM と AGM*アルゴリズムを比較した場合，$|I|$ が大きくなると AGM の支持度計算で抽出される頂点 ID の順列の数も増大する．AGM*は探索木の一部だけを探索しているため，$|I|$,$|T|$ の増加の影響を受けにくい．

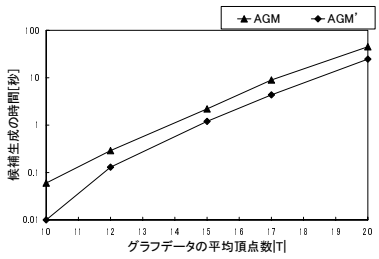
AGM*アルゴリズムの計算時間は，抽出される多頻度の数に影響を受けにくく，疎グラフデータでも有効であると考えられる．



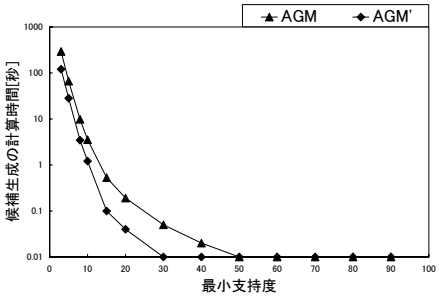
(a) $|L_E|$ v.s. 多頻度グラフの候補生成の計算時間



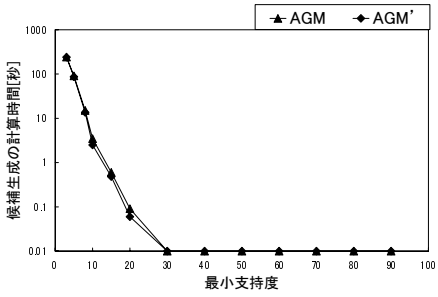
(b) $|L_V|$ v.s. 多頻度グラフの候補生成の計算時間



(c) $|T|$ v.s. 多頻度グラフの候補生成の計算時間

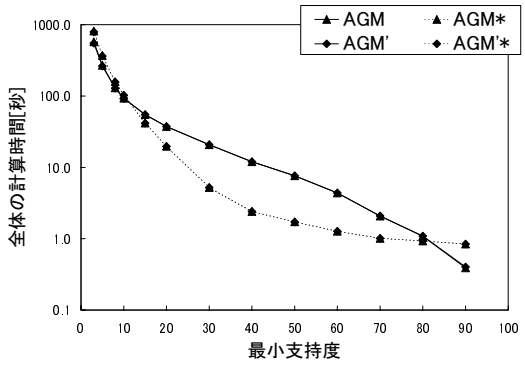


(d) $p = 1\%$ の minsup v.s. 多頻度グラフの候補生成の計算時間

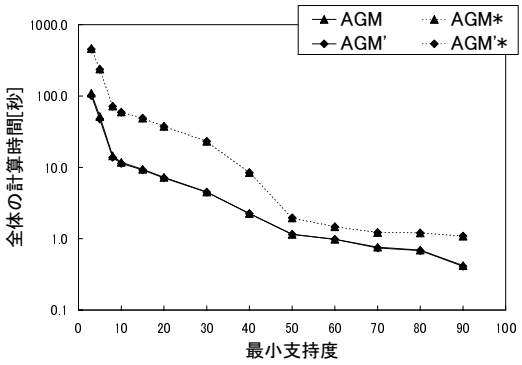


(e) $p = 30\%$ の minsup v.s. 多頻度グラフの候補生成の計算時間

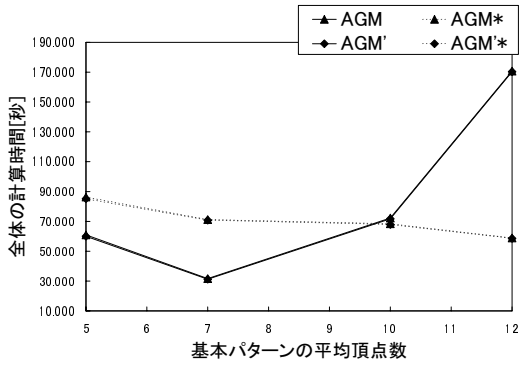
図 5: AGM と AGM' の多頻度グラフの候補生成の計算時間



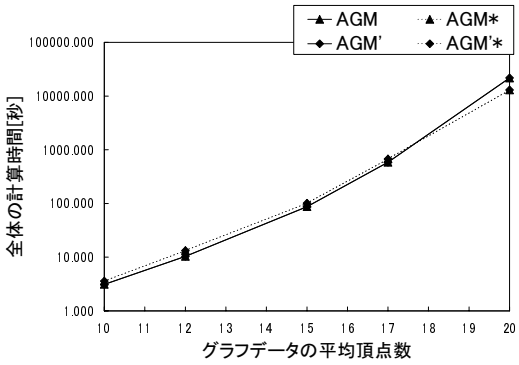
(a) $p = 1\%$ の minsup v.s. 全体の計算時間



(b) $p = 30\%$ の minsup v.s. 全体の計算時間



(c) $|I|$ v.s. 全体の計算時間



(d) $|T|$ v.s. 全体の計算時間

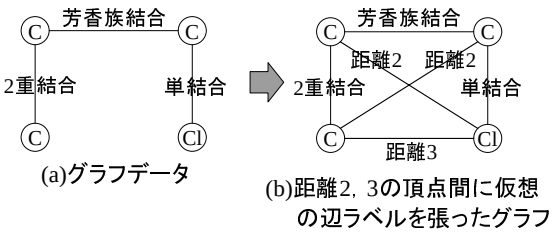
図 6: 各アルゴリズムの計算時間

4.3 応用例

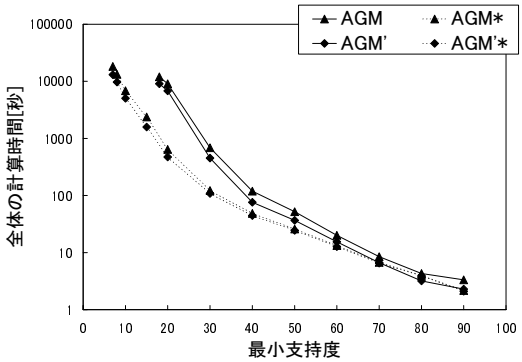
実際のデータに対する応用例として，PAKDD2000 Workshop(KDD Challenge 2000)[PAKDD 00]で提供された変異原性のデータを使用した．このデータは 230 個の化合物からなり，化合物を構成する原子の種類は 10 種類，結合の種類は 4 種類である．原子，結合，原子の種類，結合の種類をそれぞれグラフの頂点，辺，頂点ラベル，辺ラベルとして扱う．1 つのグラフデータに含まれる頂点数は，平均で約 17.6 ，最大のもので 28 であった．頂点間に辺が存在する確率は約 12.4%であり，疎グラフのデータであった．

また，化学構造データは疎グラフであるため，頂点間に辺が存在しない場合が多い．そこで頂点間の距離が 2 から 15 の頂点間には仮想的な辺ラベルを付け加えた．仮想的な辺ラベルのつけ方の例を図 7(a) にしめす．辺が存在しない頂点間に距離が 2，3 であるという仮想的なラベルを張る．すなわち，単結合，芳香族結合，距離 2，距離 3 などのラベルを持つ辺が存在することになる．仮想の辺ラベルをつけることで，頂点間の辺ラベルを特定する条件が厳しくなるため，計算時間を短縮できる．

最小支持度 $minsup$ を変化させた時のシステム全体の計算時間の結果を図 7(b) に示す．AGM* アルゴリズムの計算時間は AGM', AGM と比べて遥かに高速化されており， AGM' アルゴリズムは AGM の計算時間を 20%程度高速化していることがわかる．また，AGM' と AGM* を併用した AGM'* アルゴリズムが一番高速であることがわかる．



(a) 仮想の辺ラベル



(b) 化学構造データ (PAKDD00) の結果

図 7: 化学構造データ

5. むすび

本稿では，多頻度グラフの候補生成を高速化する AGM' アルゴリズム，支持度計算を高速化する AGM* アルゴリズム，両方のアルゴリズムを併用した AGM'* アルゴリズムについて人工データと実データによる性能評価と特徴分析を行った．その結果，AGM' と AGM* アルゴリズムは疎グラフデータに対して有効であること，実データに対しては AGM'* アルゴリズムが最も高速であることを示した．

参考文献

[Agrawal 94] Agrawal, R. and Srikant, R.: Fast Algorithms for Mining Association Rules, in Bocca, J. B., Jarke, M., and Zaniolo, C. eds., *Proc. of the 20th Very Large Data Bases Conference*, pp. 487–499, Morgan Kaufmann (1994).

[Inokuchi 00] Inokuchi, A., Washio, T., and Motoda, H.: An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data, in *Proc. of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 13–23 (2000).

[PAKDD 00] http://www. slab. dnj. ynu. ac. jp/ challenge2000/.

[西村 02] 西村 芳男, 鷲尾 隆, 吉田 哲也, 元田 浩, 猪口 明博：Apriori-based Graph Mining アルゴリズムの高速化, 第 128 回 情報処理学会 知能と複雑系研究会 (2002).

[猪口 01] 猪口 明博, 鷲尾 隆, 元田 浩：Apriori-Based Graph Mining アルゴリズムの効率化, 第 15 回人工知能学会全国大会 (2001).

[猪口 02] 猪口 明博, 鷲尾 隆, 西村 芳男, 元田 浩：グラフ構造データからの連結多頻度グラフ抽出手法, 第 16 回人工知能学会全国大会 (2002).