5P-05

ネットワークパターンマイニングアルゴリズムの効率的実装

戸崎友斗 † 武藤敦子 † 森山甲一 † 犬塚信博 †
名古屋工業大学大学院工学研究科情報工学専攻 †

1 はじめに

大規模なデータの中から有用な知識を見つけ出す手法としてデータマイニングがある．その中で，複数のテーブルからなるデータベースを対象と関係型データマイニング [1] が存在する．これは，帰納論理プログラミングの枠組みで研究されてきた．

特に，社会ネットワークを扱うネットワークパターンマイニングアルゴリズムとして Hanabi[2] が提案されている．これは，Apriori 性を用いる効率的なパターンの生成やマッチング方法を変えることで高速化 [3] が図られている．しかし，膨大な数のパターンを生成することで計算量が多くなり, 実際のデータでは計算ができない．中でも， 複雑なパターンが支持度の計算に長い時間かかってしまうという問題がある． 本研究では，ネットワークとパターンのマッチングにおいて単一化させる順序を決定することで，支持度の計算を早くすることを目的とする．

2 Hanabi

Hanabi では，関係データベース表 1 が入力として与えられる．単項の述語のリテラルを目標リテラルという．2 項の述語は，対象間のネットワークを表現し，前者と後者の引数をそれぞれ入力引数, 出力引数という．

定義 1 (近傍) データベース D のある目標リテラル e の**近傍**は，次の (1)(2) で定まるリテラル集合である．
(1) 入力引数が e である D 内のリテラルは e の近傍である．このとき，その出力項を近傍基礎項という．
(2) 全ての引数が e の近傍基礎項である D 内のリテラルは近傍である．

定義 2 (変数化) ある基礎のリテラル集合 C_g に対して，次の条件を満たすリテラル集合 C_v を C_g の変数化という．
(1)C_v は，基礎項を含まない
(2)$C_g = C_v\theta$ を満たす代入 $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$ で，$t_1, \ldots, t_n$ が全て異なる項となる θ が存在する．

ある e を頭部とし，その近傍を本体とした節を変数化したものを基本パターンという．例えば，表 1 の person1 から抽出される基本パターンは，$\mathrm{m}(x_1) \leftarrow \mathrm{f}(x_1, \boldsymbol{x_2}), \mathrm{f}(x_1, \boldsymbol{x_3}), \mathrm{f}(x_1, \boldsymbol{x_4}), \mathrm{f}(x_2, x_4), \mathrm{f}(x_3, x_4)$. である．本体の出力変数を連結変数という．

定義 3 (支持度) ネットワークを表すデータベース D におけるパターン C の支持度は次の $sup_(C)$ で表す．

$$sup_(C) = \frac{|\{t \in T | match(D, C, t)\}|}{|T|}$$

T は D の目標リテラルの集合である．支持度が最小サポートを超えるパターンを頻出パターンとする．

定義 4 (OI 包摂) 花火節 C，データベース D，$t \in D$ について，(1)$t = \mathrm{head}(C)\theta$, $\mathrm{body}(C)\theta\rho \subseteq D$，(2)$\theta = \{A_1/b_1\}$, $\rho = \{A_2/b_2, \ldots, A_n/b_n\}$，(3)$i \neq j$ ならば $b_i \neq b_j$，これらを全て満たす代入 θ,ρ があるとき，C は t にマッチするという．

定義 5 (パターン木) データベース D の全基本パターンの集合を $\mathcal{U}$ とするとき，$\mathcal{U}$ のパターン木は次の通り．
(1)$P \in \mathcal{U}$ に対し， $\mathcal{U}$ の連結変数の個数だけ葉を持つ高さ 1 のラベル付き順序木 T は $\mathcal{U}$ のパターン木である．ただし T の根ノードは P 自身を，葉は P の連結変数をラベルとする．T は P を表す．
(2) $\mathcal{U}$ のパターン木 T と $P \in \mathcal{U}$ について，T のある葉 X のラベルを P に変更し，X に P を表すパターンを置換した木 T' もパターン木である．T' は T が表すパターン P_T と P に対し，$P_T \cup body(P)\theta$ を表す．ここで $\theta = \{x/head(P)\}$，x は X の元のラベルである．これを T に P を連結するという．

3 提案手法

従来手法では，パターンとネットワークのマッチングに多く時間がかかる問題がある．また，これまでパターンとネットワークのマッチングにおいて単一化させる順序が考えられていなかった．そこで，本研究ではこの順序を決定させる方法を提案する．

Efficient implementation of network pattern mining algorithm
Yuto Tosaki†, Atsuko Mutoh†, Koichi Moriyama†and Nobuhiro Inuzuka†
†Department of Computer Science, Graduate School of Engineering, Nagoya Institute of Technology, Nagoya

表 1: 友人関係ネットワークに関するデータベース

m(X) (member)	f(X, Y) (friend)	
person1	person1	person2
person2	person1	person3
person3	person1	person4
person4	person2	person4
person5	person3	person4
person6	...	...

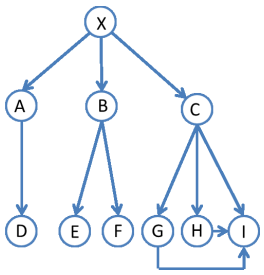

図 1: パターン例

図 1 のようなパターンは，m(X) ← f(X, A), f(X, B), f(X, C), f(A, D), f(B, E),f(B, F), f(C, G), f(C, H),f(C, I), f(G, I), f(H, I). で表される．エッジを表現している f(X,Y) の入力引数が親ノード，出力引数が子ノードである．まず，本体部の先頭からマッチングを行い，単一化させる前に本体部を並び替える．並び替えのアルゴリズムを図 2 に示す．

この重みを付けた順序であれば，他の対象と関係を多く持つノードが引数となるリテラルからマッチングを行うため，ネットワークとマッチしにくい部分がより先に調べられマッチング回数を減らすことができ，計算量を削減することができると考えられる．エッジの重みの降順と昇順のそれぞれを深さ優先探索と幅優先探索の 4 つの並び替えで比較する．

4 実験

エッジの重みの降順と昇順のそれぞれを深さ優先探索と幅優先探索の 4 つの並び替えで比較する．実験環境は，Windows 7 Professional 64-bit,CPU Intel Core i5 @3.30GHz, メインメモリ 8GB,SWI-Prolog 7.6.4 で実装したアルゴリズムを実行した．実験ではランダムに生成したネットワークを用いた．ネットワークのノー

```
Psort(P):
  input  :パターンP=head<-body;
  output :ソートしたパターンPs;
  1. body_s := ∅;
  2. for each f(A,B) ∈ body do;
  3.    w(f(A,B)) = |{f(X,Y) ∈body | X = B ∪ Y = B}|;
  4. body_w := bodyをwでソート;
  5. queue NL := headのノードから深さ優先探索した順にノードを格納;
  6. While NL ≠ ∅ do
  7.    N := NL dequeue;
  8.    body_sに{f(X,Y) ∈ body_w | X = N}を後ろから追加;
  9. return Ps := head <- body_s;
```

図 2: パターン並び替えアルゴリズム

表 2: call 回数

木の高さ	降順，深さ	昇順，深さ	降順，幅	昇順，幅
1	2246	66878	2413	120053
2	57797	445682	47122	1743461
3	1512701	3839141	516015	7048416
4	2397232		3083488	

表 3: 実行時間 (s)

木の高さ	降順，深さ	昇順，深さ	降順，幅	昇順，幅
1	0.37	0.70	0.37	1.00
2	0.67	3.12	0.62	11.79
3	17.89	28.17	6.13	70.83
4	25.47		42.58	

ド数は 50，エッジ数は 238 である．このネットワークの基本パターンをランダムに繋げて作られたパターンの木の高さが 1 から 4 までそれぞれ 18 パターンに対してパターンマッチングを行った．実験の結果を表 2, 3 に示す．call 回数は，18 パターンの単一化を行う回数の平均を表している．実行時間は，18 パターンの平均の実行時間を表している．また，木の高さが 4 のときのエッジの重みを昇順にした並び替えでは，実行時間が膨大であり空欄とする．

結果として，エッジの重みを降順とする他の対象と関係を多く持つノードからマッチングを行った方が call 回数が少なく，時間が速くなることでエッジの重みが影響することが確認できた．しかしながら，深さ優先探索と幅優先探索において，パターンによって大きく変わってしまうことからどちらが最適か確認できなかった．

5 まとめ

本研究では，データベースとパターンのマッチングにおいて単一化させる順序をエッジに重みを付けることで決定し，並び替えの方法を比較した．エッジの重みが，call 回数や実行時間に影響することが確認できた．今後の課題としては，より多く実験をしてパターンの特徴を考え，パターンの特徴に応じて単一化順序を変更することが今後の課題である．

参考文献

[1] Luc Dehaspe and Hannu Toivonen. Discovery of frequent datalog patterns. *Data Min. Knowl. Discov.*, Vol. 3, No. 1, pp. 7–36, 1999.

[2] 西尾 典晃, 犬塚 信博. 開いた構造を持つ事例を対象とした関係的知識発見. 第 75 回情処全国大会, 2013.

[3] 森 僚太, 犬塚 信博. OI 包摂に基づくネットワークパターンマイニングアルゴリズムの効率的実装. 第 80 回情処全国大会, 2018.