4A1-1

系列二分決定グラフを用いた頻出部分グラフの圧縮表現

Compressed Representation of Frequent Subgraphs Using Sequence BDDs

岡崎　文哉 *1　奥山　葉月 *2　瀧川　一学 *1*3　湊　真一 *1
Fumiya Okazaki　Hazuki Okuyama　Ichigaku Takigawa　Shin-ichi Minato

*1北海道大学大学院情報科学研究科
Graduate School of Information Science and Technology, Hokkaido Universit

*2北海道大学工学部
School of Engineering, Hokkaido University

*3科学技術振興機構，さきがけ
JST,PRESTO

Frequent subgraph mining is a problem of listing frequent subgraphs in a given set of graphs. Often the output subgraphs are of huge number, and for post-processing those mining results, it needs an efficient way to store and index them. To address this problem, we propose a novel methods to enumerate and index the subgraph outputs. Our idea is to canonically represent frequent subgraphs as edge sequences, and store them in Sequence Binary Decision Diagrams (SeqBDDs). Three different sequence-encoding of subgraphs are proposed, and their performance are evaluated by numerical experiments.

1. はじめに

データマイニングの主要な研究課題の 1 つである頻出パターンマイニングは，与えられたデータ集合中に一定頻度以上で現れるパターンをすべて列挙する問題である．その対象には，アイテムセットや系列データ，時系列，グラフ，木など様々なものがあげられる．頻出部分グラフマイニングは，グラフデータから部分グラフを列挙する問題である．低分子化合物の構造式 [Takigawa 13] や RNA 二次構造 [Karklin 05]，ネットワークなど多種多様なものがグラフで表現され，グラフデータ中に頻出する部分グラフを発見する問題は，化合物データベースやタンパク質-タンパク質ネットワーク，XML 文節データベースなどのグラフデータベースから特徴的な部分構造を見つける数々の応用が存在する．そのため，これまでにデータマイニングの分野では，いくつものグラフマイニングアルゴリズムが提案されている (FSG[Kuramochi 01]，AGM[Inokuchi 00, Inokuchi 03]，gSpan[Yan 02]，Gaston[Nijssen 04])．

頻出部分グラフマイニングでは，列挙対象となる部分グラフが膨大な数出力されることが多いため，マイニング結果である列挙したパターン集合の保存や再利用の計算コストが大きくなる．列挙した結果のうち，ある制約を満たすものだけを取り出したい場合，列挙に対して制約を与え，制約を満たす部分グラフのみを列挙すればよい．しかし，現実的にデータの利活用を考えると，制約が変わるたびに計算コストの大きい列挙を行うことは現実的ではなく，列挙結果をはじめから索引化しておきたい．こうした列挙結果の再利用の需要は大きく，例えば，化合物データベースがグラフデータとして与えられ，頻出する部分グラフのうち，大きさが制限された部分グラフを取り出したい場合や，異なる化合物データベースに出現する部分構造を削除したい場合などが考えられる．この目的の達成を考えると，ただグラフ情報を保存しておくだけでは，不十分である．なぜなら，異なるデータベースに出現する部分グラフを削除したい場合，同じグラフ構造は一意の形になるように保存しておく必要があり，そうでなければ，2 つのグラフが同じであるか判定する問題，グラフ同型性判定問題を解く必要がある．また例に挙げたような様々な再利用を考えた場合，扱うグラフ数は数百万以上になることも多く，情報を効率的に圧縮して保存し，また索引化されている必要がある．

このような背景から，頻出パターンマイニングにおいて，アイテム集合や系列集合を対象とした問題に対して出力パターンを圧縮索引化して保持する研究が行われてきた．頻出アイテム集合マイニングでは，出力をアイテム集合を扱うデータ構造である ZDD[Minato 01] に格納して出力する手法 [Minato 08] があり，頻出文字列集合マイニングでは，系列集合を扱うデータ構造である系列二分決定グラフ (SeqBDD)[Loekito 10] に格納し出力する手法 [Denzumi 13] が提案されている．ZDD や SeqBDD では，巨大なパターン集合を圧縮するのみならず，圧縮した表現のまま，パターン集合同士の差集合や積集合など多様な代数演算が利用でき，効率的なマイニング結果の活用が可能である．一方，本研究の対象であるグラフマイニングでは，こうした有効なパターン集合の圧縮索引化の方法が未だ確立しておらず，効率的な手法の構築が望まれてきた．

本研究では，部分グラフ集合を系列集合として表現することで，頻出部分グラフマイニングの出力グラフを，SeqBDD を用いて効率よく列挙索引化する手法を提案する．グラフマイニングの出力である部分グラフパターンがエッジの系列として一意に表すことができることを利用することで，SeqBDD による圧縮索引化の利用，および，SeqBDD 同士の代数演算体系に基づくマイニング結果の効率的利活用が期待できる．部分グラフを系列として表現するために，頻出部分グラフマイニングの既存手法である gSpan で用いられる最小 DFS コードに基づいて，頂点や辺にラベルが付与されたグラフをエッジの系列として表現する手法を 3 種類提案する．さらに提案する 3 種類の手法に対して SeqBDD による圧縮索引化を行い，実データを用いて実験的に比較する．また，圧縮という観点のみの比較として，代表的な手法であるコンパクトトライによる圧縮結果との比較実験を行う．

連絡先: 岡崎　文哉，北海道大学大学院情報科学研究科，fokazaki@ist.hokudai.ac.jp

2. 準備

2.1 頻出部分グラフマイニングと最小支持度

本論文で扱うグラフとは，多重辺や自己ループを許さず，頂点や辺におけるラベル付無向グラフとする．

頻出部分グラフマイニングとは，グラフデータセット $GS = \{G_i \mid i = 1, 2, 3, \ldots, n\}$ に対して，閾値 σ が与えられた時，ある部分グラフ g の支持度 (以下，support) support(g) =

$|\{G_i \mid G_i \in GS, g \text{ が } G_i \text{のある部分グラフに同型 }\}|$ について $\mathrm{support}(g) \geq \sigma$ となる部分グラフ g を全列挙するという問題である．この与えられる閾値 σ を最小支持度 (以下，minsup) と呼ぶ．

2.2 DFS コードと最小 DFS コード [Yan 02]

本研究では，頻出部分グラフマイニングの既存手法である gSpan[Yan 02] で用いられているグラフの表現方法である DFS コードに基づき，グラフをエッジの系列として表現する．この節では，DFS コードと最小 DFS コードについて説明する．

DFS コードとは，グラフ G を頂点に 0 から番号をつけながら深さ優先探索をした場合に得られる DFS 木に対して，そのエッジを到達順に並べたものである．エッジの一つ一つは，2 つの頂点番号と 2 つの頂点のラベルと辺のラベル $(i, j, \mathrm{label}_i, \mathrm{edgelabel}_{i,j}, \mathrm{label}_j)$ の 5 項組で表される．このとき未探索の頂点へ向かう辺を前進辺 (forward edge)，すでに探索済みの頂点に向かう辺を後進辺 (backward edge) と呼び，E^f を G の前進辺の集合，E^b を G の後進辺の集合とする．G の深さ優先探索順 1 つに対して DFS コードが一意に決まり，さらに DFS コードに対して辞書的順序 [Yan 02] を与えることで，辞書順最小になる DFS コードを最小 DFS コードとしグラフを一意に表現することができる．

gSpan では，深さ優先探索順に部分グラフを列挙するとき，DFS コードが最小になるものだけを列挙することで同じグラフを 2 度探索しないアルゴリズムである．

2.3 系列二分決定グラフ (SeqBDD)[Loekito 10]

SeqBDD は，系列集合を表現するデータ構造である．図 1 に系列集合 {aaa，aba，bbc，bc} を表す SeqBDD の例を示す．SeqBDD は 1 つの根節点と 0 または 1 の終端節点を持ち，それ以外の各接点は 2 値のラベル付けに対応した 0-枝，1-枝と呼ばれる 2 つの出力枝を持つ．根節点から終端節点までのパスが一つの系列を表現しており，1 終端にたどり着くパスがその系列集合に含まれる系列であることを表現している．SeqBDD は，単に系列集合の圧縮表現のみならず，SeqBDD 同士の和集合や積集合などを圧縮した構造のまま行う代数系を備えており，その後の利活用のための索引化手法にもなっている．

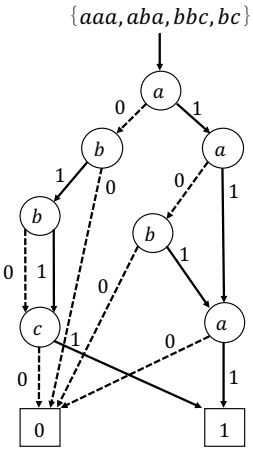

図 1: SeqBDD の例

3. 提案手法

3.1 概要

本研究のアイデアは，gSpan アルゴリズムの列挙結果である各部分グラフが自然に最小 DFS コードとして出力できることに着目し，これらを系列集合出力とみなして SeqBDD に格納するというものである．最小 DFS コードを用いることで，部分グラフの一意性が保証できる．次節では，SeqBDD へ部分グラフを格納するための最小 DFS コードの系列表現として，3 種類の手法を提案する．

3.2 グラフの系列表現

3.2.1 直接的な系列表現 (手法 1)

DFS コードを系列に対応させる直接的な手法を提案する．DFS コードはグラフのエッジを並べたものである．そこで，5 項組で表される 1 つのエッジを 1 文字と対応させることで，部分グラフを系列として表現する．すでに出現したエッジと文字の対応を保存するハッシュ構造を持ち，初めて出現するエッジに対しては新しい文字と対応させ，ハッシュ構造を更新する．これは，提案する残り 2 種類の手法にも共通する手続きである．

3.2.2 非冗長化 DFS コードによる系列表現 (手法 2)

2 つ目の手法として，非冗長化 DFS コードを提案する．

グラフ G の DFS コード $\alpha = (\alpha_1, \ldots, \alpha_m), \alpha_n = (i_n, j_n, \mathrm{label}_{i_n}, \mathrm{edgelabel}_{i_n,j_n}, \mathrm{label}_{j_n})$ が与えられたとき，G の非冗長化 DFS コード $\beta = (\beta_0, \beta_1, \ldots, \beta_m)$ は，

- $n = 0$ のとき
  $\beta_0 = \mathrm{label}_{i_0}$
- $1 \leq n \leq m$ のとき
  $\beta_n = (v, \mathrm{edgelabel}, \mathrm{nodelabel})$
  $$= \begin{cases} (i_n, \mathrm{edgelabel}_{i_n,j_n}, \mathrm{label}_{j_n}) & (\alpha_n \in E^f) \\ (j_n, \mathrm{edgelabel}_{i_n,j_n}, -1) & (\alpha_n \in E^b) \end{cases}$$

非冗長化 DFS コードは，DFS コードから冗長な情報を取り除いたものである．DFS コードを生成するとき，新しい頂点につける頂点番号は，一意に定まり，DFS コードに出現する頂点のラベルで既出のものも存在する．よって，エッジが前進辺である場合は，新しく追加する頂点番号は，保存しておく必要が無い．また，すでに存在する頂点のラベルも保存する必要がない．よってそれらの情報を取り除き，5 項組で表されていた DFS コードを 3 項組と最初のノードラベルで表したものが非冗長化 DFS コードである．

非冗長化 DFS コードによる系列表現を行う場合，まず，各グラフの DFS コードに対して非冗長化 DFS コードを生成し，手法 1 で行った手続きと同様に系列に対応させ SeqBDD へ格納する．

3.2.3 相対 DFS コードによる系列表現 (手法 3)

3 つ目の手法として，相対 DFS コードを提案する．相対 DFS コードを定義するために，最右パスを説明する．

最右パスは，与えられた DFS コードの深さ優先順で，開始点から一番新しく伸ばした頂点までのパスである．最右パスは前進辺のみから成る．この時，そのグラフから 1 つエッジを伸ばして得られるグラフが最小 DFS コードであるためには，最右パス上の頂点から前進辺を伸ばすか，最右パス上の頂点に戻る後進辺である必要がある．よって，新しく追加する辺は，最右パスに対して相対的に記憶することで表現できる．そこで，最右パスに含まれる頂点数を d としたとき，最右パスに最後に追加した頂点から順に $0, \ldots, d-1$ として相対順序を定める．

最右パス上の頂点に対して定めた相対順序を用いて，相対 DFS コードを以下のように定義する．

グラフ G の非冗長化 DFS コード $\alpha = (\alpha_0, \ldots, \alpha_m)$ が与えられたとき，G の相対 DFS コード $\beta = (\beta_0, \ldots, \beta_m)$ は，

- $\alpha_0 = \beta_0$

- $\alpha_n = (v, \text{edgelabel}, \text{nodelabel})$ に対して，非冗長化 DFS コード $\alpha_0, \ldots, \alpha_{n-1}$ の最右パス上の v に対応する相対順序を v' として $\beta_n = (v', \text{edgelabel}, \text{nodelabel})$ とする．

また，相対 DFS コードによる系列表現を行う場合，まず，非冗長化 DFS コードに対して相対 DFS コードを生成し，手法 1 で行った手続きと同様に系列と対応させ SeqBDD に格納する．

4. 計算機実験

3 節で述べた 3 種類の系列表現を用いた提案手法の評価として，頻出部分グラフマイニングの出力に対する SeqBDD のサイズの比較や計算時間に対して実験を行う．本研究で用いた実験環境は，CPU:Intel(R) Core(TM) i7-3770 3.40GHz，メモリ:7.9GiB である．

4.1 使用したデータセット

今回，ベンチマークとしてよく用いられる Mutag と CPDB の 2 つのデータを使用し，実験を行った．Mutag はグラフ数 188 で平均ノード数 17.9，平均エッジ数 19.8，CPDB はグラフ数 684 で平均ノード数 14.1，平均エッジ数 14.6 のデータセットである．

表 1 に Mutag，CPDB それぞれに対して，最小支持度を変化させたときの頻出部分グラフ数，DFS コードの 5 項組を 1 文字としたときの総文字数を示す．

	Mutag		CPDB	
minsup(%)	系列数	総文字数	系列数	総文字数
50	366	2564	13	35
30	6743	71825	48	184
20	74790	1017148	100	441
15	416125	6593443	139	659
10	1980916	34616403	345	2005
8	3397816	60675967	492	2904
6	8066081	151602886	851	5779

表 1: Mutag，CPDB 最小支持度に対する系列数，総文字数

手法 1 では，DFS コードをそのまま符号化するため，総エッジ数がそのまま総文字数になる．また，手法 2，手法 3 では系列の先頭にラベルを持つため，系列数 + 総文字数 が総文字数になる．

4.2 SeqBDD のサイズに関する実験

SeqBDD に系列を格納する際に，2 種類の変数順序が考えられる．SeqBDD の先頭に系列の先頭がくる変数順序 1 と SeqBDD の先頭に系列の最後がくる変数順序 2 が考えられ，どちらの変数順序を用いることで SeqBDD のサイズが抑えられるかは，系列集合の傾向によって異なる．表 2 に，3 種類の手法による SeqBDD の節点数を比較した実験結果に加え，変数順序 1，2 に対してそれぞれ SeqBDD を生成した場合の実験結果を示す．

表の文字種類は，系列集合に出現する文字の種類数を指す．実験結果より，手法 1 に対して手法 2，さらに手法 3 で出現する文字の種類数が減少している．これは，部分グラフを相対 DFS コードで表現した効果であると考えられる．また，3 種類の手法の比較をすると，ラベルの種類数，節点数ともに手法 1 よりも手法 2，そして手法 3 のほうが減少していることが多いことがわかる．しかし，CPDB の minsup50%を見てわかるように，節点数は必ずしも減少するわけではないことがわかる．これは手法 2，手法 3 では，系列の長さが手法 1 に比べ 1 だけ大きいためである．そのため，系列の数が多くなると，手法 2，手法 3 の系列表現の効果が顕著に現れていることがわかる．これは，系列に出現する文字の種類が大きく減少したためであると考えられる．

	Mutag			CPDB		
変数順序		1	2		1	2
minsup(%)	文字種類	節点数	節点数	文字種類	節点数	節点数
手法 1:直接的な手法						
50	83	248	389	11	11	17
30	196	2854	4459	24	41	53
20	338	26119	41744	36	77	91
15	457	142688	226087	48	105	137
10	593	539673	841898	77	248	341
8	641	754136	1220927	93	351	478
6	743	1878239	3150979	119	540	744
手法 2:非冗長化 DFS コード						
50	42	226	348	10	13	18
30	67	2678	4298	17	43	56
20	97	20883	39478	21	78	92
15	120	109139	212758	28	102	135
10	144	362783	795117	38	231	337
8	149	514291	1123671	44	337	470
6	168	1455577	2859992	51	508	725
手法 3:相対 DFS コード						
50	15	133	251	7	13	19
30	21	1746	3258	11	40	59
20	33	16483	30663	15	63	75
15	36	86095	152712	19	88	129
10	46	241429	447093	27	210	306
8	51	338909	669436	31	291	436
6	63	1109522	2073812	34	414	721

表 2: Mutag，CPDB 手法 3 種類の比較と変数順序による比較実験

また，変数順序 1 が変数順序 2 に対して節点数が小さくなることがわかった．これは DFS コードの性質上，系列の先頭が多く共有できるためであると考えられる．変数順序 1 が有効であるという結果を得たため，SeqBDD とトライの比較は，変数順序 1 を用いて行う．

4.3 SeqBDD とトライに関する比較実験

さらに，SeqBDD の圧縮度を評価するため，コンパクトトライによる圧縮結果との比較実験を行った．トライ (Trie) は多数の文字列データを辞書順に分類・整理した木構造による索引であり，SeqBDD は，トライに含まれる等価な部分木を全て共有したものに対応づけられることが知られている．そこで，トライと SeqBDD の記憶量を比較することで，共有による圧縮の効果を評価することができる．

SeqBDD とトライの比較実験の結果を表 3 に示す．本研究では，分岐のない節点をまとめることで節点を圧縮して保持するコンパクトトライを用いた．実験結果より，minsup が 50%や 30%のとき，SeqBDD とトライでは，節点数に差はあまりなく，CPDB ではトライのほうが小さくなることもあることがわかる．しかし，minsup を下げると，データ構造間の差が大きくなり，SeqBDD のほうが，大幅に小さく表現されている事がわかる．また，トライでは，3 種類の手法による接点数の差異はない．これは，トライが接頭部分しか共有しないデータ構造であるからである．それに加えて，SeqBDD では，手法 3 が最も小さく効率的に表現できている．

4.4 計算時間に関する実験

提案した 3 種類の手法に対する計算時間を比較する実験を行った．表 4 に実験結果を示す．用いたデータは Mutag で，系列数を比較的多いもので比較するため最小支持度は 10%とした．

実験結果より，いずれの手法も系列を生成する時間に対して，SeqBDD を生成する時間により計算時間がかかるという結果となった．また，系列生成，SeqBDD 生成ともに手法 1

	Mutag		CPDB	
データ構造	SeqBDD	トライ	SeqBDD	トライ
minsup(%)	節点数	節点数	節点数	節点数
手法 1:直接的な手法				
50	248	260	11	10
30	2854	4946	41	32
20	26119	52454	77	64
15	142688	282635	105	90
10	539673	1343178	248	246
8	754136	2284747	351	346
6	1878239	5480149	540	575
手法 2:非冗長化 DFS コード				
50	226	261	13	11
30	2678	4947	43	34
20	20883	52455	78	66
15	109139	282636	102	93
10	362783	1343179	231	249
8	514291	2284748	337	349
6	1455577	5480150	508	578
手法 3:相対 DFS コード				
50	133	261	13	11
30	1746	4947	40	34
20	16483	52455	63	66
15	86095	282636	88	93
10	241429	1343179	210	249
8	338909	2284748	291	349
6	1109522	5480150	414	578

表 3: Mutag，CPDB 手法 3 種類に対する SeqBDD とトライの比較実験

	手法 1	手法 2	手法 3
系列生成	12.069s	10.934s	8.94s
SeqBDD 生成	16.357s	16.184s	13.688s

表 4: 3 種類の手法に対する計算時間比較 (Mutag，最小支持度 10%)

より手法 2，手法 2 より手法 3 のほうが計算時間が抑えられることがわかる．

系列を生成するにあたって，手法 1 より手法 2，手法 2 より手法 3 が計算時間が短いのは，文字に対応するコードを記憶しておく配列に原因があると考えられる．実際に生成されたコードに対して，対応する文字がすでに存在するのか，あるいは新しい文字に対応させるのかを判別するときに，配列の大きさ，つまり，ラベルの種類数が，大きくなる手法ほど，時間がかかると考えられる．

5. まとめ

本稿では，頻出部分グラフマイニングにおける出力部分グラフを系列として表現し，系列集合を扱うデータ構造である SeqBDD を用いて圧縮索引化を行う手法を提案した．頂点や辺にラベルが付与されている部分グラフを系列として表現する方法を 3 種類提案し，SeqBDD のサイズや計算時間に対する比較実験により，頻出部分グラフを圧縮索引化するために SeqBDD を用いた場合，手法 3:相対 DFS コードがメモリ効率や計算時間という点で効率的であるという結果を得た．

今後の展望としては，さらに SeqBDD を小さくするような符号化や SeqBDD の構築の高速化，SeqBDD の演算を用いた実データに対する応用等が考えられる．

謝辞

本研究は JSPS 科研費 26330242，16K13852 および JST さきがけの助成を受けたものです．

参考文献

[Denzumi 13] Denzumi, S., Tsuda, K., Arimura, H., and Minato, S.: Compact Complete Inverted Files for Texts and Directed Acyclic Graphs Based on Sequence Binary Decision Diagrams, in *Proceedings of the Prague Stringology Conference 2013, Prague, Czech Republic, September 2-4, 2013*, pp. 157–167 (2013)

[Inokuchi 00] Inokuchi, A., Washio, T., and Motoda, H.: An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data, in *Principles of Data Mining and Knowledge Discovery, 4th European Conference, PKDD 2000, Lyon, France, September 13-16, 2000, Proceedings*, pp. 13–23 (2000)

[Inokuchi 03] Inokuchi, A., Washio, T., and Motoda, H.: Complete Mining of Frequent Patterns from Graphs: Mining Graph Data, *Machine Learning*, Vol. 50, No. 3, pp. 321–354 (2003)

[Karklin 05] Karklin, Y., Meraz, R. F., and Holbrook, S. R.: Classification of Non-Coding RNA Using Graph Representations of Secondary Structure, in *Biocomputing 2005, Proceedings of the Pacific Symposium, Hawaii, USA, 4-8 January 2005*, pp. 5–16 (2005)

[Kuramochi 01] Kuramochi, M. and Karypis, G.: Frequent Subgraph Discovery, in *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, pp. 313–320 (2001)

[Loekito 10] Loekito, E., Bailey, J., and Pei, J.: A binary decision diagram based approach for mining frequent subsequences, *Knowl. Inf. Syst.*, Vol. 24, No. 2, pp. 235–268 (2010)

[Minato 01] Minato, S.: Zero-suppressed BDDs and their applications, *STTT*, Vol. 3, No. 2, pp. 156–170 (2001)

[Minato 08] Minato, S., Uno, T., and Arimura, H.: LCM over ZBDDs: Fast Generation of Very Large-Scale Frequent Itemsets Using a Compact Graph-Based Representation, in *Advances in Knowledge Discovery and Data Mining, 12th Pacific-Asia Conference, PAKDD 2008, Osaka, Japan, May 20-23, 2008 Proceedings*, pp. 234–246 (2008)

[Nijssen 04] Nijssen, S. and Kok, J. N.: A quickstart in frequent structure mining can make a difference, in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pp. 647–652 (2004)

[Takigawa 13] Takigawa, I. and Mamitsuka, H.: Graph mining: procedure, application to drug discovery and recent advances, *Drug Discovery Today*, Vol. 18, No. 12, pp. 50 – 57 (2013)

[Yan 02] Yan, X. and Han, J.: gSpan: Graph-Based Substructure Pattern Mining, in *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pp. 721–724 (2002)