DEIM Forum 2011 B10-2

逆探索法によるグラフ系列マイニングの高速化

生田　泰章† 　猪口　明博† 　鷲尾　隆†

† 大阪大学 産業科学研究所　〒 567–0047 大阪府茨木市美穂ヶ丘 8-1
E-mail: †{ikuta,inokuchi,washio}@ar.sanken.osaka-u.ac.jp

あらまし　人間関係ネットワークは人が頂点，関係が辺であるグラフで表現でき，人がネットワークに参加，脱退することで頂点や辺が増減する．遺伝子とその相互作用からなる遺伝子ネットワークも同様であり，これらのネットワークの変化はグラフ系列により表現できる．グラフ系列から頻出部分系列をマイニングする手法である GTRACE は，計算効率の点で課題が残されている．その課題を克服するため本稿では逆探索法の原理を用い，GTRACE を更に高速化するための手法を提案し，性能評価した結果を報告する．

キーワード　グラフ系列マイニング，逆探索法

Efficient Graph Sequence Mining using Reverse Search

Hiroaki IKUTA†, Akihiro INOKUCHI†, and Takashi WASHIO†

† The Institute of Scientific and Industrial Research Osaka University
8-1 Mihogaoka, Ibaraki, Osaka, 567 Japan
E-mail: †{ikuta,inokuchi,washio}@ar.sanken.osaka-u.ac.jp

1. はじめに

情報技術の発展は著しく，膨大なデータを蓄えることが可能となった．しかし，蓄積された大規模なデータから人手で有用な情報を発見することは，非常に困難である．このような背景から，大規模なデータから分析者にとって有用な知識の発見を行う，データマイニングの研究が盛んに行われている．近年特に構造を持つデータの蓄積が多く行われ，その中でも，頂点と辺で構成されるグラフは，最も一般的な構造の 1 つである．頻出グラフマイニングは，グラフで表されたデータ集合から，頻出する部分グラフを発見する手法であり，これまで盛んな研究が行われてきた．

しかし，現実世界では構造は時間の経過と共に変化することが多い．例えば，Web のネットワーク構造は時間を追うごとに変化している．Web サイトが新規に作成，もしくは削除されたり，サイト間でリンクが発生，もしくは削除されたりすることで，構造は変化する．従来の頻出グラフマイニングでは構造の変化まで扱うことができなかったが，近年，GTRACE(Graph Transformation Sequence Mining) [7] が提案され，グラフ構造が変化しているデータから，頻出な部分構造の変化をマイニングすることが可能となった．GTRACE は，グラフ構造の変化をグラフの系列として考え，このグラフ系列をマイニングすることによって，大規模なグラフ系列に埋もれた，頻繁に現れる構造の変化 (頻出部分グラフ系列) を発見する．

GTRACE は従来の静的なグラフ構造ではなく，動的なグラフ構造 (グラフ系列) を扱うことが可能なアルゴリズムであるが，不要な頻出部分グラフ系列も出力するため，効率性の観点から改良の余地が残されている．そこで，本稿ではグラフ系列をマイニングする新たな手法を提案する．

2. GTRACE

2.1 グラフ系列の表現

本研究が対象とするグラフ系列は，ラベル付きグラフがステップを経るごとに変化する場合を考える．ラベル付きグラフ g を頂点集合 V，辺集合 E，ラベル集合 L，ラベル付け関数 f を用い，$g = (V, E, L, f)$ という四つ組で表す．ここで頂点集合，辺集合はそれぞれ $V = \{v_1, \cdots, v_n\}$，$E = \{(v, v') \mid (v, v') \in V \times V\}$ であり，ラベル集合 L が $f : V \cup E \to L$ と定義されるグラフである．図 1(a) は観測されたグラフ系列の例である．$g^{(j)}$ $(j \in \mathbb{N})$ はグラフ系列中において j 番目のグラフである．GTRACE では観測されたグラフ系列に対して，以下の 2 つの仮定を置く．

- 連続する 2 つのグラフ $g^{(j)}$ と $g^{(j+1)}$ の間では，ごく一部の構造のみが変化する．
- 各グラフは疎グラフである．

例えば人間関係ネットワーク (各個人が頂点，個人間の関係が辺に対応) を考えたとき，人間関係が急に大きく変わることは考え難く，さらに各個人が他の大多数の人と関係を有しているとも考え難い．つまり，上記の仮定があてはまる．その他，実

表 1　グラフ系列データの変換規則

頂点追加 $vi^{(j,k)}_{[u,l]}$	ラベルが l，頂点 ID が u である頂点を $g^{(j,k)}$ へ追加し，$g^{(j,k+1)}$ へ変換
頂点削除 $vd^{(j,k)}_{[u,\bullet]}$	頂点 ID が u である頂点を $g^{(j,k)}$ から削除し $g^{(j,k+1)}$ へ変換
頂点ラベル変更 $vr^{(j,k)}_{[u,l]}$	頂点 ID が u である頂点のラベルを l に変更し，$g^{(j,k)}$ を $g^{(j,k+1)}$ へ変換
辺追加 $ei^{(j,k)}_{[(u_1,u_2),l]}$	頂点 ID が u_1 と u_2 である頂点間にラベル l の辺を追加し，$g^{(j,k)}$ を $g^{(j,k+1)}$ へ変換
辺削除 $ed^{(j,k)}_{[(u_1,u_2),\bullet]}$	頂点 ID が u_1 と u_2 である頂点間から辺を削除し，$g^{(j,k)}$ を $g^{(j,k+1)}$ へ変換
辺ラベル変更 $er^{(j,k)}_{[(u_1,u_2),l]}$	頂点 ID が u_1 と u_2 である頂点間の辺のラベルを l へ変更し，$g^{(j,k)}$ を $g^{(j,k+1)}$ へ変換

頂点削除と辺削除の変換規則，vd と ed は頂点 ID の指定のみで変換可能なので，引数 l はダミー変数であり，‘$\bullet$’ で表す．$u_1 \leqq u_2$ を必ず満たす．

世界の多くの構造の変化はこれらの仮定を満たしていると考えられる．

観測されたグラフ系列を $d = \langle g^{(1)} g^{(2)} g^{(3)} \cdots g^{(n)} \rangle$ と表す．ここで $g^{(1)}$ は系列の先頭を，$g^{(n)}$ は系列の末尾を表している．また，1 つのグラフ系列 d にはある有限な頂点集合 V_d が存在し，d 中の各グラフ $g^{(j)}$ の頂点集合 $V(g^{(j)})$ は常に V_d に含まれる ($V(g^{(j)}) \subseteqq V_d$) と仮定する．さらに，1 つのグラフ系列中の各頂点 $v \in V_d$ には個別の頂点 ID $id(v)$ を付与する．系列中の j 番目のグラフ $g^{(j)}$ について，頂点 ID に関する 2 つの集合を以下のように定義する．

$$ID_V(d) = \{id(v) \mid v \in V(g^{(j)}), g^{(j)} \in d\}$$
$$ID_E(d) = \{(id(v), id(v')) \mid (v, v') \in E(g^{(j)}), g^{(j)} \in d\}$$

ID_V は $g^{(j)}$ の頂点集合における頂点 ID の集合を，ID_E は $g^{(j)}$ の辺集合において各辺に対応する頂点 ID の組の集合である．

また，GTRACE では観測されたグラフ系列中の連続する 2 つのグラフ $g^{(j)}$ と $g^{(j+1)}$ を人工的に補完する．観測されたグラフ系列 d の各グラフ $g^{(j)}$ を外部状態，補完するグラフ系列 s の各グラフ $g^{(j,k)}$ を内部状態と呼ぶ．

GTRACE では，最小単位の変換を変換規則を用いて表す．

定義 1. $g^{(j,k)}$ を $g^{(j,k+1)}$ へ変換する変換規則を $tr^{(j,k)}_{[o_{jk},l_{jk}]}$ で表す．ただし，以下の 3 点を満たす．

- tr は頂点や辺の追加，削除，ラベルの変更のいずれか．
- o_{jk} は変換される頂点や辺について，$ID_V(d)$，あるいは $ID_E(d)$ の要素である．
- $l_{jk} \in L$ は変換される頂点や辺のラベルである．　■

ここでは，簡単化のため変換規則 $tr^{(j,k)}_{[o_{jk},l_{jk}]}$ を $tr^{(j,k)}_{[o,l]}$ と略記する．GTRACE では，表 1 に示す 6 つの変換規則を定義し，グラフ間の差異を表現している．

以上より，変換系列を以下のように定義する．

定義 2. 内部状態系列 $s = \langle g^{(j,1)} g^{(j,2)} g^{(j,3)} \cdots g^{(j,m_j)} \rangle$ を変換規則を用いて $seq(s^{(j)}) = \langle tr^{(j,1)}_{[o,l]} tr^{(j,2)}_{[o,l]} tr^{(j,3)}_{[o,l]} \cdots tr^{(j,m_j-1)}_{[o,l]} \rangle$ と表し，内部状態変換系列と呼ぶ．さらに，外部状態系列 $d = \langle g^{(1)} g^{(2)} g^{(3)} \cdots g^{(n)} \rangle$ を内部状態変換系列の系列である変換系列 $seq(d) = \langle seq(s^{(1)}) seq(s^{(2)}) seq(s^{(3)}) \cdots seq(s^{(n-1)}) \rangle$ で表す．　■

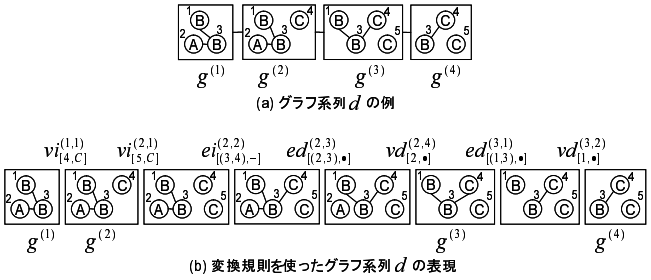

図 1　変換規則を用いたグラフ系列の例

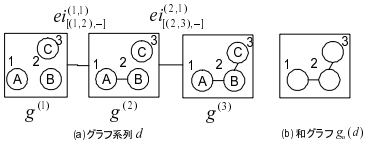

図 2　和グラフの例

変換系列はグラフが徐々に変化するという仮定の下で，2 つの状態間の差異を変換規則を用いて表しており，グラフ表現を直接用いた系列の表現よりも簡潔である．また，どのようなグラフ系列も表 1 における 6 つの変換規則で表現することが可能である [7].

外部状態の順序は観測されたグラフ系列中のグラフの順序と同一であるが，内部状態の順序は人工的に補完されたグラフ系列であるため，表し方は様々なものが考えられる．GTRACE では計算コスト，空間コストを抑えるため，グラフの編集距離に基づき最短の補完系列を選択する [1]．それでもなお，内部状態変換系列は系列内の変換規則の順序が一意に決まることは無い．そこで，変換規則間の線形順序を定義し，変換規則はその順序に従って並んでいるものとする．

定義 3. 2 つの変換規則 $tr1^{(j,k)}_{[o,l]}$ と $tr2^{(j',k')}_{[o',l']}$ について，以下の条件を満すとき，内部状態変換系列において $tr1^{(j,k)}_{[o,l]}$ を $tr2^{(j',k')}_{[o',l']}$ より前に配置し，これを $tr1^{(j,k)}_{[o,l]} \prec_{TR} tr2^{(j',k')}_{[o',l']}$ と表す．

$$tr1^{(j,k)}_{[o,l]} \prec_{TR} tr2^{(j',k')}_{[o',l']}$$
$$= \begin{cases} true\ if\ \ j < j' \\ true\ if\ \ tr1 \prec_{tr} tr2\ and\ j = j' \\ true\ if\ \ o \prec_{obj} o',\ tr1 = tr2\ and\ j = j' \\ true\ if\ \ l < l',\ o =_{obj} o',\ tr1 = tr2\ and\ j = j' \end{cases}$$
■

線形順序 $tr1 \prec_{tr} tr2$ は許容性定理 [7] に従う線形順序であり以下のように定義する．

定義 4. 2 つの変換規則 $tr1^{(j,k)}_{[o,l]}$ と $tr2^{(j',k')}_{[o',l']}$ が与えられたとき，$tr1, tr2 \in \{vi, vr, vd, ei, er, ed\}$ であり，変換規則の種類は，$vi \prec_{tr} ei \prec_{tr} vr \prec_{tr} er \prec_{tr} ed \prec_{tr} vd$ の順序に従う．　■

また，$o \prec_{obj} o'$ は以下の定義に従う．

定義 5. 2 つの変換規則 $tr1^{(j,k)}_{[o,l]}$ と $tr2^{(j',k')}_{[o',l']}$ について，線形順序 $o \preceq_{obj} o'$ は以下の条件を満たす．ただし，$tr1, tr2 \in \{ei, ed, er\}$ のとき，$o = (u_1, u2)$，$o' = (u'_1, u'_2)$ という頂点のペアを表すこととする．

$o \preceq_{obj} o'$

$$= \begin{cases} true\ if\ \ o \leqq o',\ tr1 \in \{vi, vd, vr\}\ and\ tr2 \in \{vi, vd, vr\} \\ true\ if\ \ u_1 \leqq u_1',\ tr1 \in \{ei, ed, er\}\ and\ tr2 \in \{ei, ed, er\} \\ true\ if\ \ u_2 \leqq u_2',\ u_1 = u_1',\ tr1 \in \{ei, ed, er\} \\ and\ tr2 \in \{ei, ed, er\} \end{cases} \blacksquare$$

以上の定義により，変換規則間の順序が決まる．以降では全ての変換系列においてはこれらの順序付けに従い，変換規則が昇順にソートされているとする．例えば，図 1(a) のグラフ系列 d は，図 1(b) のように変換規則を用いて表すことができる．グラフ系列 d は $seq(d) = \langle vi^{(1,1)}_{[4,C]} vi^{(2,1)}_{[5,C]} ei^{(2,2)}_{[(3,4),-]} ed^{(2,3)}_{[(2,3),\bullet]} vd^{(2,4)}_{[2,\bullet]} ed^{(3,1)}_{[(1,3),\bullet]} vd^{(3,2)}_{[1,\bullet]} \rangle$ とコンパイルされる．なお，“–” は辺ラベルを表す．

2.2 rFTS(relevant Frequent Transformation Sub-sequences) マイニング

GTRACE は与えられたグラフ系列の集合から，後述する rFTS(relevant Frequent Transformation Sub-sequences) をマイニングするアルゴリズムである．頻出部分変換系列 (FTS) は，最小支持度 σ' 以上の支持度を持つ部分変換系列のことを指す．グラフ系列の集合 $DB = \{\langle tid, d\rangle | d = \langle g^{(1)} \cdots g^{(n)} \rangle\}$ が与えられたとき, 部分変換系列 $seq(d')$ の支持度 $\sigma(seq(d'))$ は次の式で表される．

$$\sigma(seq(d')) = |\{tid | \langle tid, d\rangle \in DB,\ seq(d') \sqsubseteq seq(d)\}|$$

なお，$seq(d') \sqsubseteq seq(d)$ の定義は文献 [7] を参照されたい．以下に rFTS について例を用いて説明する．

図 1 を人間関係を表すグラフ系列 (頂点：人，辺：関係) の集合からマイニングされた FTS のグラフ系列とする．この図の辺のつながりに注目すると，頂点 ID 1 と頂点 ID 2，頂点 ID 2 と頂点 ID 3 は直接関係がある．また，頂点 ID 1 と頂点 ID 3 は頂点 ID 2 の人物を通して関係がある．しかし，頂点 ID 4 と頂点 ID 5 の人物はそれぞれ，他のどの人物とも関係がない．この例では関連のない頂点は 2 点のみだが，入力として与えられる DB のグラフ系列の頂点 ID 数が多ければ，1 つの FTS の中には関連のない頂点が多数含まれる可能性がある．そこで，GTRACE がマイニングする FTS は，このような関連のない頂点が含まれていない FTS を対象とする．この FTS を関連のある FTS(relavant-FTS: rFTS) と呼ぶ．FTS が rFTS かどうかを判断するため，以下に和グラフを定義する．

定義 6. 変換系列 $seq(d)$ に対応する和グラフ $g_u(seq(d))$ を以下のように定義する．

$$g_u(seq(d)) = (V_u, E_u)$$
$$V_u = \{u \mid tr^{(j,k)}_{[u,l]} \in seq(d), tr \in \{vi, vd, vr\}\}$$
$$\cup \{u, u' \mid tr^{(j,k)}_{[(u,u'),l]} \in seq(d), tr \in \{ei, ed, er\}\}$$
$$E_u = \{(u, u') \mid tr^{(j,k)}_{[(u,u'),l]} \in seq(d), tr \in \{ei, ed, er\}\} \quad \blacksquare$$

例 1. 図 *2(a)* に示される変換系列 $\langle ei^{(1,1)}_{[(1,2),-]} ei^{(2,1)}_{[(2,3),-]} \rangle$ の和グラフは図 *2(b)* と同型である．

変換系列 $seq(d)$ の和グラフが連結であるとき，変換系列

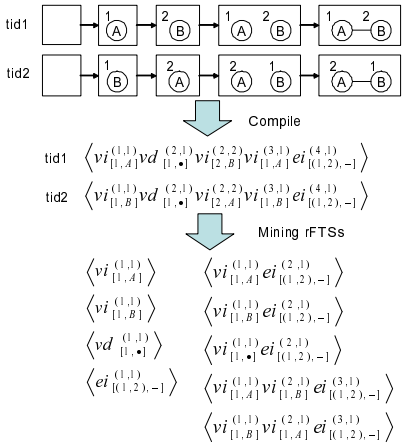

図 3　DB から rFTS をマイニングする手順例

$seq(d_1) = \langle vi^{(1,1)}_{[1,A]} \rangle$
$seq(d_2) = \langle vi^{(1,1)}_{[1,A]} vi^{(2,1)}_{[2,B]} \rangle$
$seq(d_3) = \langle vi^{(1,1)}_{[1,A]} vi^{(2,1)}_{[2,B]} vi^{(3,1)}_{[3,C]} \rangle$
$seq(d_4) = \langle vi^{(1,1)}_{[1,A]} vi^{(2,1)}_{[2,B]} vi^{(3,1)}_{[3,C]} ei^{(4,1)}_{[(1,2),-]} \rangle$
$seq(d_5) = \langle vi^{(1,1)}_{[1,A]} vi^{(2,1)}_{[2,B]} vi^{(3,1)}_{[3,C]} ei^{(4,1)}_{[(1,2),-]} ei^{(4,2)}_{[(2,3),-]} \rangle$
$seq(d_6) = \langle vi^{(1,1)}_{[1,A]} vi^{(2,1)}_{[2,B]} vi^{(3,1)}_{[3,C]} ei^{(4,1)}_{[(1,2),-]} ei^{(4,2)}_{[(2,3),-]} ed^{(5,1)}_{[(2,3),-]} \rangle$

図 4　GTRACE のマイニング手順の詳細

$seq(d)$ の頂点 ID は互いに関連があると定義する．以上より，GTRACE が対象とする問題を以下に示す．

問題 1. グラフ系列の集合 $DB = \{\langle tid, d\rangle | d = \langle g^{(1)} \cdots g^{(n)} \rangle\}$, 最小支持度 σ' を入力として，DB から σ' 以上の支持度を持つ全ての関連のある頻出部分変換系列 *(rFTS)* を列挙する．

例 2. 図 *3* は 2 つのグラフ系列で構成される DB と最小支持度 $\sigma' = 2$ を入力として，*9* つの *rFTS* をマイニングする *GTRACE* の手順の例である．グラフ系列を変換系列に変換し，最小支持度以上の *FTS* を列挙する．その後和グラフが連結グラフの *FTS* のみを出力する．この例では，$\langle vi^{(1,1)}_{[1,A]} vi^{(2,1)}_{(2,B)} \rangle$ と $\langle vi^{(1,1)}_{[1,B]} vi^{(2,1)}_{[2,A]} \rangle$ は *FTS* であるが，和グラフが連結グラフでないため，出力されない．

rFTS を効率よくマイニングするために，GTRACE は (1) 入力グラフ系列の和グラフの集合から頻出連結部分グラフを探索し，(2) 各頻出連結部分グラフの和グラフが同型である rFTS を列挙する，という 2 ステップの操作を行う．2 ステップ目の詳細は，まず，アイテム集合の系列から頻出する部分系列 (パターン) をマイニングする手法である PrefixSpan [4] と同様に，FTS の末尾に変換規則を 1 つ追加し新たな FTS を再帰的に列挙することを行う．そして，FTS を全て列挙した後，列挙された FTS の和グラフが連結かどうかの確認を行い，連結なもののみを rFTS として出力する．

例 3. 図 *4* は *GTRACE* アルゴリズムにおいて *rFTS* がマイニングされる手順の内 *2* ステップ目の途中経過を示している．*GTRACE* は *FTS* である $seq(d_i)$ $(i \in \mathbb{N})$ をマイニングした後，マイニングされた *FTS* の末尾に変換規則を *1* つ追加し最小支

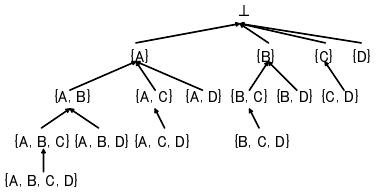

図 5　木で表現された探索空間の例

持度以上の *FTS* $seq(d_{i+1})$ をマイニングする操作を再帰的に行う．全ての *FTS* が列挙された後，$seq(d_2)$，$seq(d_3)$，$seq(d_4)$ は和グラフが非連結のため後処理で削除される．

2.3　既存アルゴリズムの問題点

前節で GTRACE の対象問題，マイニング手順について述べた．GTRACE は 1 ステップ目で rFTS となり得る和グラフを探索するが，2 ステップ目で直接 rFTS をマイニングせず，FTS をマイニングした後に後処理で和グラフが連結でない FTS を削除し，rFTS を列挙する．そのため，GTRACE は 2 ステップ目において，rFTS をマイニングする以前に，多数の FTS をマイニングする必要があるので非効率である．例えば，図 4 に示される $seq(d_6)$ は和グラフが連結グラフのため rFTS である．しかし，$seq(d_6)$ がマイニングされるまでに和グラフが連結グラフではない $seq(d_2)$，$seq(d_3)$，$seq(d_4)$ がマイニングされる．そこで，rFTS のみをマイニングする新たなアルゴリズムを提案し，計算効率の向上を図る．

3.　提 案 手 法

3.1　逆探索に基づく rFTS マイニング

全ての rFTS を効率良くマイニングするために，逆探索の原理 [2] を用いて rFTS のみをマイニングする手法を提案する．逆探索は解の列挙問題に対する効率的なアルゴリズムの構築を可能にする手法である．S を解集合としたとき，逆探索では $x \in S$ が唯一の親 $P(x) \in S$ を持つ親子関係を示す関数 $P : S \to S$ を定義する．ここで，x は唯一の親を持つので，探索空間は S の要素を節点，親子関係を枝とする木 T で表現される．そして，この T を根より深さ優先に探索することで，S の要素を重複なくかつ完全に探索することが可能となる．

アイテム集合 $I = \{A, B, C, D\}$ の列挙問題について逆探索の例を図 5 に示す．親子関係を示す関数 P をアイテム集合 $x = \{i_1, i_2, \cdots, i_{k-1}, i_k\}$ の最後のアイテム i_k を削除し，アイテム集合 $x' = \{i_1, i_2, \cdots, i_{k-1}\}$ を得る関数と定義したとき，x の親が $P(x)$ である探索空間は図 5 の列挙木のように表される．根から深さ優先に探索する (木の矢印とは逆方向に進む) ことによって，全てのアイテム集合を列挙することが可能となる．

第 2.3 節で述べたように，rFTS である変換系列 $seq(d_i)$ に対する $P(sed(d_i))$ を，$sed(d_i)$ の末尾の変換規則を削除して得られる変換系列 $seq(d_{i-1})$ と定義したとき，支持度の逆単調性より，$sed(d_{i-1})$ が FTS であることは保証される．しかしながら，rFTS であることは保証されない．

そこで，本稿では，rFTS のみを列挙するために，$P(seq(d_i))$ が rFTS であるような関数 P を定義する．ここで，S を正準形で表された全ての rFTS の集合とする．正準形とは各 rFTS が一意に識別されるための条件を具備した形をしていることを指す．変換系列に関する正準形の定義を以下に示す．

定義 7. 同型な変換系列 $sed(d), seq(d')$ について，変換規則が後述する関数 $P = \{P_1, P_2, P_3\}$ に基づいて，$(tr_k, tr_{k-1}, \cdots, tr_1)$，$(tr'_k, tr'_{k-1}, \cdots, tr'_1)$ の順にそれぞれ削除されるとする．このとき，$seq(d), seq(d')$ のコードをそれぞれ $\alpha = (tr_1, tr_2, \cdots, tr_k)$，$\beta = (tr'_1, tr'_2, \cdots, tr'_h)$ と定義する．*2* つのコードについて，以下の条件の内どちらかを満たせば，α は β よりも小さい，あるいは等しいとし，$\alpha \preceq \beta$ と表す．

- t, q について，$1 \leqq t \leqq min(k, h), q < t$ のとき，$tr_t \prec_{TR} tr'_t$ かつ任意の q について $tr_q = tr'_q$ を満たす t が存在する．
- q, k について，$0 \leqq q \leqq k, k \leqq h$ のとき，任意の t について $tr_q = tr'_q$ である．

同型な変換系列に対応するコードの内，最小のコードを正準形と定義する．■

定義 7 より，変換規則を削除する順序は，後述する関数 $P = \{P_1, P_2, P_3\}$ に基づいて決められる．以下，それぞれの関数の詳細を述べる．

はじめに，第 1 関数 P_1 を以下のように定義する．

定義 8. 関数 P_1 を，*rFTS* である $seq(d) \in S$ の頂点に関する変換規則 $\{vi, vd, vr\}$ の内，一番最後の変換規則を削除し，変換系列 $seq(d')$ を得る関数と定義する．■

関数 P_1 より次の補題が得られる．なお，下記の変換系列の長さとは，変換系列に含まれる変換規則の数である．

補題 1. $seq(d) \in S$ を頂点に関する変換規則を含む長さ *1* 以上の *rFTS* とする．このとき，$g_u(P_1(seq(d)))$ は $g_u(seq(d))$ と同型である．■

証明. $seq(d)$ は rFTS であるため，$seq(d)$ の和グラフ $g_u(seq(d))$ は連結である．もし関数 P_1 によって削除される $seq(d)$ の変換規則が $tr^{(j,k)}_{[u,l]}$ であれば，$seq(d)$ に頂点 ID u が端点である辺に関する変換規則が必ず存在する．これは $seq(d)$ の和グラフが連結であることに由来する．従って，頂点 ID が u である頂点を端点にもつ辺に作用する変換規則が存在するため，$tr^{(j,k)}_{[u,l]}$ を削除した後でも頂点 u は和グラフ $g_u(P_1(seq(d)))$ に存在する．従って，$g_u(P_1(seq(d)))$ は $g_u(seq(d))$ と同型となる．□

補題 1 より rFTS である $seq(d)$ の和グラフと $P_1(seq(d))$ が同型なので，$P_1(seq(d))$ は関連のある変換系列である．また，$P_1(seq(d))$ は $seq(d)$ の部分系列でもあるので，支持度の逆単調性から頻出である．つまり，$P_1(seq(d))$ は rFTS である．さらに，頂点に関する一番最後の変換規則を 1 つ削除するので，$seq(d)$ の唯一の親である．そして，S の要素である $seq(d)$ は正準形であるので，正準形の定義より $P_1(seq(d))$ も正準形である．後述する関数 P_2, P_3 についても同様のことが言える．なお，頂点に関する変換規則を含む長さ 1 の rFTS $seq(d) \in S$ が

与えられたとき，$P_1(seq(d)) = \langle\rangle$ とする．

例 4. 図 *4* における $seq(d_6)$ について，$P_1(seq(d_6))$ と $P_1(P_1(seq(d_6)))$ はそれぞれ以下のようになる．

$$P_1(seq(d_6)) = \langle vi^{(1,1)}_{[1,A]} vi^{(2,1)}_{[2,B]} ei^{(3,1)}_{[(1,2),-]} ei^{(3,2)}_{[(2,3),-]} ed^{(4,1)}_{[(2,3),-]} \rangle$$
$$P_1(P_1(seq(d_6))) = \langle vi^{(1,1)}_{[1,A]} ei^{(2,1)}_{[(1,2),-]} ei^{(2,2)}_{[(2,3),-]} ed^{(3,1)}_{[(2,3),-]} \rangle$$

$seq(d_6)$, $P_1(seq(d_6))$, $P_1(P_1(seq(d_6)))$ の和グラフは図 *2(b)* となり，同型である．

続いて，第 2 関数 P_2 を以下のように定義する．

定義 9. 頂点に関する変換規則を含まない *rFTS* である $seq(d) \in S$ が与えられたとき，関数 P_2 は，一番最後の変換規則 $tr^{(j,k)}_{[o,l]}$ を削除し，$seq(d')$ を得る関数である．ただし，変換規則 $tr^{(j,k)}_{[o,l]}$ を削除できるのは $o = o'$ かつ $j' < j$ なる変換規則 $tr^{(j',k')}_{[o',l']}$ が存在する場合に限る． ■

もし，$seq(d)$ が頂点の変換規則を含んでいる場合，第 1 関数 P_1 を複数回適用し，頂点に関する変換規則を全て削除した後，頂点に関する変換規則がない rFTS $seq(d')$ について，第 2 関数 P_2 を適用することが可能である．さらに，P_2 は $seq(d')$ の長さが和グラフ $g_u(seq(d))$ の辺の数よりも大きい場合に適用することが可能となる．これは，$seq(d')$ の少なくとも 1 つの変換規則が $g_u(seq(d))$ の各辺に対応するからである．

補題 2. 頂点に関する変換規則が含まれておらず，長さが自身の和グラフの辺の数 $|E(g_u(seq(d)))|$ よりも大きい *rFTS* である $seq(d) \in S$ が与えられたとき，$g_u(P_2(seq(d)))$ は $g_u(seq(d))$ と同型である． ■

証明. $seq(d)$ は rFTS のため，和グラフ $g_u(seq(d))$ は連結である．また，$seq(d)$ には辺に関する変換規則しか含まれていない．関数 P_2 によって削除される $seq(d)$ の変換規則が $tr^{(j,k)}_{[(u_1,u_2),l]}$ であったとき，$(u_1,\ u_2)$ に作用する変換規則は $seq(d)$ 内に必ず存在する．これは，P_2 が適用することができるのは $(u_1, u_2) = (u'_1, u'_2)$ かつ $j' < j$ なる変換規則 $tr^{(j',k')}_{[(u'_1,u'_2),l']}$ が存在する場合に限られることに由来する．つまり，$tr^{(j,k)}_{[(u_1,u_2),l]}$ を削除した後でも和グラフ $g_u(P_2(seq(d)))$ には頂点 u_1，u_2 とその間に辺が存在する．従って，$g_u(seq(d))$ と同型となる． □

補題 2 により，$P_2(seq(d))$ は常に関連がある変換系列となり，支持度の逆単調性より頻出な変換系列でもある．つまり，$P_2(seq(d))$ は rFTS であり，辺に関する一番最後の変換規則を削除するので，$seq(d)$ の唯一の親である．

例 5. $seq(d'_3) = \langle ei^{(1,1)}_{[(1,2),-]} ei^{(1,2)}_{[(2,3),-]} ei^{(2,1)}_{[(2,3),-]} \rangle$ が与えられたとき，$P_2(seq(d'_3)) = \langle ei^{(1,1)}_{[(1,2),-]} ei^{(1,2)}_{[(2,3),-]} \rangle$ となる．定義 *9* より $P_2(seq(d'_3))$ を関数 P_2 の引数とすることはできない．

最後に，第 3 関数 P_3 を以下のように定義する．この定義の中で，グラフの切断点とはグラフの頂点の内，その頂点を削除したときグラフの連結成分が増えるものをいう [3].

定義 10. 頂点に関する変換規則を含まず，和グラフ $g_u(seq(d))$ の辺の数と同じ長さの *rFTS* である $seq(d) \in S$ が与えられたとき，関数 P_3 を一番最後の変換規則 $tr^{(j,k)}_{[o,l]}$ を削除し $seq(d')$ を得る関数と定義する．ただし，$tr^{(j,k)}_{[o,l]}$ が削除できるのは，$o = (u, u')$ の内，u, u' が $g_u(seq(d))$ において共に切断点でない場合に限る． ■

もし，rFTS である $seq(d)$ の長さが和グラフ $g_u(seq(d))$ の辺の数と異なっていれば，第 1 関数 P_1 もしくは第 2 関数 P_2 を複数回適用することで，和グラフ $g_u(seq(d'))$ の辺の数と同じ長さの rFTS $seq(d')$ を得ることができ，$seq(d')$ には関数 P_3 を適用することが可能である．また，定義 10 より，和グラフの辺の数と同じ長さの rFTS $seq(d)$ について，$P_3(seq(d))$ は $g_u(P_3(seq(d)))$ が連結グラフであることより，常に関連のある変換系列となる．さらに，$P_3(seq(d))$ は支持度の逆単調性より常に頻出な変換系列でもある．従って，$P_3(seq(d))$ は rFTS であり，辺に関する変換規則の一番最後を 1 つ削除するので，$seq(d)$ の唯一の親である．

図 6 は rFTS に関数 P_1，P_2，P_3 を適用することにより，唯一の rFTS を返す親子関係の例を表している．$seq(d_6)$ は，頂点に関する変換規則が 3 個含まれていることから，関数 P_1 を 3 回適用することができ，$seq(d'_5)$，$seq(d'_4)$，$seq(d'_3)$ の順に rFTS が得られる．次に，$seq(d'_3)$ には頂点 ID のペア $(2,\ 3)$ を結ぶ辺の変換規則が 2 個含まれているので，関数 P_2 を 1 回適用することができ，$seq(d'_2)$ を得ることができる．最後に，変換規則が空になるまで，$seq(d'_2)$ に関数 P_3 を適用することができ，$seq(d'_1)$ が得られる．この例では $seq(d_6)$ から $seq(d'_2)$ までは和グラフは同型であり，連結グラフである．また，$seq(d'_1)$ の和グラフも連結グラフである．

前述の 3 つの関数，P_1，P_2，P_3 を解集合 S に適用することによって構築された探索空間 T は，$\langle\rangle$ を根とした木によって表される．この探索空間 T を根から深さ優先探索することで，正準形の rFTS のみを完全に列挙することが可能である．ここで，変換系列 $seq(d)$ について $P_1(seq(d)), P_2(seq(d)), P_3(seq(d))$ という唯一の親が得られる一方，これらの親は $seq(d)$ 以外にも複数の子を持つ．そこで，親である変換系列 $seq(d)$ から子である変換系列の集合を得る関数を以下に定義し，それらの関数を $P_1^{-1}(seq(d)), P_2^{-1}(seq(d)), P_3^{-1}(seq(d))$ と書く．なお，関数 P_i^{-1} は関数 P_i の逆写像ではないことに注意されたい．(注1)

定義 11. 変換系列 $seq(d)$ の子となる変換系列を $seq(d')$ としたとき，関数 $P_i^{-1}(i = 1, 2, 3)$ は以下の式に従う．

$$P_i^{-1}(seq(d)) = \{seq(d') \in S | P_i(seq(d')) = seq(d)\}$$ ■

以上より，探索は関数 $P^{-1} = \{P_1^{-1}, P_2^{-1}, P_3^{-1}\}$ に従い，rFTS に変換規則を 1 つずつ追加することで進める．従って，前述の親子関係を示す関数 $P = \{P_1, P_2, P_3\}$ の適用で，木で表現された S の探索空間を，重複無くかつ完全に rFTS を探索することが可能となる．

(注1)：一般に関数 f の逆関数は，f が全単射であるときにのみ定義される．P_i は全単射でないので，P_i^{-1} は P_i の逆関数ではない．

$$
\begin{array}{lll}
P_3(seq(d'_1)) = \langle\rangle & & \\
seq(d'_1) = P_3(seq(d'_2)) = \langle & ei^{(1,1)}_{[(1,2),-]}\rangle & \in P_3^{-1}(\perp) \\
seq(d'_2) = P_2(seq(d'_3)) = \langle & ei^{(1,1)}_{[(1,2),-]} ei^{(1,2)}_{[(2,3),-]}\rangle & \in P_3^{-1}(seq(d'_1)) \\
seq(d'_3) = P_1(seq(d'_4)) = \langle & ei^{(1,1)}_{[(1,2),-]} ei^{(1,2)}_{[(2,3),-]} ed^{(2,1)}_{[(2,3),-]}\rangle & \in P_2^{-1}(seq(d'_2)) \\
seq(d'_4) = P_1(seq(d'_5)) = \langle vi^{(1,1)}_{[1,A]} & ei^{(2,1)}_{[(1,2),-]} ei^{(2,2)}_{[(2,3),-]} ed^{(3,1)}_{[(2,3),-]}\rangle & \in P_1^{-1}(seq(d'_3)) \\
seq(d'_5) = P_1(seq(d_6)) = \langle vi^{(1,1)}_{[1,A]} vi^{(2,1)}_{[2,B]} & ei^{(3,1)}_{[(1,2),-]} ei^{(3,2)}_{[(2,3),-]} ed^{(4,1)}_{[(2,3),-]}\rangle & \in P_1^{-1}(seq(d'_4)) \\
seq(d_6) = \langle vi^{(1,1)}_{[1,A]} vi^{(2,1)}_{[2,B]} vi^{(3,1)}_{[3,C]} & ei^{(4,1)}_{[(1,2),-]} ei^{(4,2)}_{[(2,3),-]} ed^{(5,1)}_{[(2,3),-]}\rangle & \in P_1^{-1}(seq(d'_5))
\end{array}
$$

図 6　rFTS の親子関係の例

入力: rFTS $seq(d_p)$, データ集合 DB, 最小支持度 σ', $i = 3$.
出力: rFTS の集合 S.

GTRACE-RS($seq(d_p)$, DB, σ', S, i)

```
1: if seq(d_p) ≠ min, then
2:     return;
3: insert seq(d_p) into S;
4: while i > 0
5:     Call Subprocedure(seq(d_p), DB, σ', S, i);
6:     i = i − 1;
7: return;
```

Subprocedure($seq(d_p)$, DB, σ', S, i)

```
1: set C to ∅;
2: scan DB,
3: find all transformation rules r
       such that seq(d_p) ◇ r ∈ P_i^{-1}(seq(d_p));
4: insert seq(d_p) ◇ r in C and count its frequency;
5: for each frequent seq(d_p) ◇ r in C do
6:     Call GTRACE-RS(seq(d_p) ◇ r, DB, σ', S, i);
7: return;
```

図 7　GTRACE-RS のアルゴリズム

3.2　アルゴリズム (GTRACE-RS)

前節において親子関係を示す関数 $P = \{P_1, P_2, P_3\}$ によって，探索空間を木で表現できることを示した．つまり，逆探索により効率よく rFTS のみを列挙できることを示した．図 7 は本稿で提案するアルゴリズムの擬似コードであり，このアルゴリズムを GTRACE-RS と呼ぶ．GTRACE-RS はグラフ系列のデータ集合 DB から最小支持度 σ' 以上の rFTS を S に蓄積し出力するアルゴリズムである．関数 $P^{-1} = \{P_1^{-1}, P_2^{-1}, P_3^{-1}\}$ に従って探索空間を深さ優先に探索を行う．図 7 の $seq(d_p) \diamond r$ は，変換規則 r を $seq(d_p)$ に追加することを意味する．ただし，$seq(d_p) \diamond r \in P_i^{-1}(seq(d_p))$ $(i = 1, 2, 3)$ になるように，変換規則 r を追加する．また，$seq(d_p) \neq min$ は，$seq(d_p)$ が正準形かどうかを確認し，正準形でない場合はそれ以降の探索を行わない．GTRACE-RS のアルゴリズムの概要は，まず頂点に関する変換規則が含まれておらず，長さが $g_u(seq(d_p))$ の辺の数と同じ rFTS である $seq(d_p)$ について，$P_3^{-1}(seq(d_p))$ により得られる rFTS を S に蓄積する．その後 $P_2^{-1}(seq(d_p))$, $P_1^{-1}(seq(d_p))$ の順に，得られた rFTS をマイニングし S に追加する．以上を再帰的に繰り返すことで，全ての rFTS を列挙し S を出力する．具体的な実装方法として，P_3^{-1} は gSpan [5] を拡張して実装を行い，P_2^{-1}，P_1^{-1} は PrefixSpan [4] を拡張して実装した．

4. 評 価 実 験

本節では GTRACE-RS と GTRACE の比較実験を行う．両手法とも C++によって実装し，比較実験のために Intel Xeon X5560 2.80GHz の CPU，4GB のメインメモリを搭載した計算機 HP Z600 を用いた．

4.1　人工データによる実験

人工データは表 2 のパラメータを用いて生成した．生成する変換系列は 2 種類あり，1 種類が $|DB|$ 個の変換系列，もう 1 種類は rFTS として埋め込むための N 個の変換系列である．以下にグラフ系列の生成手順を示す．

はじめに，初期のグラフ (初期外部状態)$g^{(1)}$ を 2 頂点間の辺存在確率 p_e の値に基づいて生成した．そして，グラフ系列の頂点 ID 数が $|V_{avg}|$ になるまで変換規則を末尾に追加した．その際，頂点や辺を追加する確率は p_i に，削除する確率は p_d に，ラベルの変更確率は $1 - p_i - p_d$ にそれぞれ従い，連続する 2 グラフの編集距離は平均 d_e の値に従うようにした．頂点，辺のラベル数はそれぞれ $|L_v|$ ，$|L_e|$ の値とした．このような過程により生成された 1 つのグラフ系列を $|DB|$ 個生成する．頻出部分系列も同様の手順で生成する．

表 3 は $|DB|$，$|V_{avg}|$，$|L_e|$，σ' のいずれかを変化させたときの実験結果である．GTRACE は後処理を行わず，FTS の列挙をしたときの計算時間と FTS の数を記載している．GTRACE-RS は計算時間と rFTS の数を記載している．また，表中の表中の “-”は，2 時間以上経っても計算が終了しなかったことを示す．表 3 から，どのパラメータの値を変化させても大幅に計算時間を短縮することが可能となった．この実験結果について，特に興味深い結果は列挙した FTS と rFTS の数の違いである．GTRACE-RS が列挙した rFTS の数は GTRACE が列挙した FTS の数よりも大幅に少ないため，それだけ無駄な探索をすることなくグラフ系列マイニングを行うことができたと言える．

さらに，個別のパラメータの変化における実験結果の考察を行う．まず，データサイズである $|DB|$ を変化させたときに GTRACE-RS の計算時間は線形に増加しており，この結果は従来の頻出パターンマイニングの実験結果と同様である．次に，変換系列の平均 ID 数 $|V_{avg}|$ を変化させたときの結果から，GTRACE-RS は GTRACE に比べより大きく長いグラフ系列に対しても，現実的な時間でマイニングの結果を得ることが可能であることが分かった．そして，辺ラベル数 L_e について，L_e の数が少ないとき，変換系列が同型である (同じ変化を示す) ものが非常に多くなるため，計算時間，FTS の数が増える結果となった．最後に最小支持度 σ' の値を小さくしていくと，GTRACE-RS の計算時間は指数関数的に増加しており，この結果は従来の頻出パターンマイニングの実験結果と同様である．

表 2　人工データ生成パラメータと最小支持度のデフォルト値

パラメータ	デフォルト値
$vi^{(j,k)}_{[o,l]}$, $ei^{(j,k)}_{[o,l]}$ が変換規則の末尾に付加される確率	$p_i = 80$
$vd^{(j,k)}_{[o,l]}$, $ed^{(j,k)}_{[o,l]}$ が変換規則の末尾に付加される確率	$p_d = 10$
FTS 内の平均 ID 数	$\|V_{avg}\| = 6$
rFTS として埋め込まれる FTS 内の平均 ID 数	$\|V'_{avg}\| = 3$
頂点ラベル数	$\|L_v\| = 5$
辺ラベル数	$\|L_e\| = 5$
rFTS として埋め込まれる FTS 内の数	$N = 10$
変換系列数	$\|DB\| = 1000$
2 頂点間の辺存在確率	$p_e = 15$
連続する外部状態の平均編集距離	$d = 2$
最小支持度	$\sigma' = 10$

表 3　人工データの実験結果

$\|DB\|$	1,000	3,000	7,000	10,000
avg. len.	42.9	44.0	43.5	43.4
PM comptime	2.0	6.5	15.0	21.4
# of rFTSs	6307	6872	6325	6266
GT comptime	217.3	951.2	2226.2	3465.3
# of FTSs	171787	190876	170387	170902
$\|V_{avg}\|$	6	8	15	20
avg. len.	42.9	61.2	135.5	188.7
PM comptime	2.0	6.3	228.7	4335.0
# of rFTSs	6307	20755	3653370	81012875
GT comp.time	217.3	3599.3	-	-
# of FTSs	171787	936115	-	-
$\|L_e\|$	1	3	7	10
avg. len.	43.5	43.7	43.8	43.0
PM comptime	54.5	5.1	1.9	0.97
# of rFTSs	70257	12972	5318	3333
GT comptime	2195.9	644.0	205.9	122.1
# of FTSs	2995499	474564	132538	62897
σ' [%]	5	7.5	10	15
avg. len.	42.9	42.9	42.9	42.9
PM comptime	377.1	30.7	2.0	1.8
# of rFTSs	90607156	5744037	6307	1630
GT comptime	2122.4	472.3	217.3	93.3
# of FTSs	176177313	11891069	171787	43100

PM: 提案手法, GT: GTRACE, comptime: 計算時間 [秒],
avg. len.: データの平均変換規則数,
# of rFTSs (or FTSs): rFTS(FTS) の数

表 4　エンロン社データのパラメータと最小支持度のデフォルト値

パラメータ	デフォルト値
変換系列の頂点 ID 数	$\|V\| = 182$
外部状態数	$n = 7$
頂点ラベル数	$\|L_v\| = 8$
辺ラベル数	$\|L_e\| = 5$
変換系列数	$\|DB\| = 123$
最小支持度	$\sigma' = 10\%$

4.2　実データによる実験

本節では，実データに対して性能評価をするために，エンロン社電子メールデータ [12] を用いて GTRACE と GTRACE-RS との比較実験を行う．このデータは 1998 年 11 月 16 日 (月) から 2001 年 3 月 25 日 (日) までの 123 週間で 182 人が電子メールのやり取りをしたデータである．データの前処理として，各人に頂点 ID を付与し，ある二人が 1 日の間に電子メールのやり取りが行われると 2 頂点間に辺を張った．また，各頂点には，役職を示す 8 種類のラベル (“CEO”，“Director”，“Employee”，“Lawyer”，“Manager”，“President”，“Trader”，“Vice President”) を付与し，電子メールのやり取りの量に合わせて 5 種類の辺ラベルを付与した．この前処理によって 1 日の電子メールのやり取りがグラフ (外部状態) として表現できる．そして，各曜日ごとにグラフ (外部状態) を作成し，図 8 のように一本のグラフ系列を作成した．従ってエンロン社の電子メールデータは，頂点 ID 数 $|V| = 182$，外部状態数 $n = 7$，頂点ラベル数 $L_v = 8$，辺ラベル数 $L_e = 5$，変換系列数 $|DB| = 123$ のグラフ系列集合として表現される．表 4 に示す，前処理によって得たグラフ系列集合に関するパラメータを変化させ，計算時間とマイニングされた FTS の数を比較する実験を行った．前節

表 5　実データの実験結果

$\|V\|$	100	140	150	182
PM comptime	0.5	2.2	15.9	278.6
# of rFTSs	33227	31391	47015	1558833
GT comptime	16.8	118.3	-	-
# of FTSs	66072	154541	-	-
σ'[%]	40	30	20	10
PM comptime	2.0	6.1	30.0	278.6
# of rFTSs	974	3548	14419	158833
GT comptime	14.4	387.3	-	-
# of FTSs	4129	29253	-	-
n	4	5	6	7
PM comptime	5.85	34.1	95.6	278.6
# of rFTSs	5542	21214	51727	158833
GT comptime	423.8	-	-	-
# of FTSs	67997	-	-	-

PM: 提案手法, GT: GTRACE, comptime: 計算時間 [秒],
# of rFTSs (or FTSs): rFTS(FTS) の数

同様，GTRACE がマイニングした FTS の数は rFTS と関連性の無い FTS の総和であり，GTRACE-RS がマイニングした FTS の数は rFTS の数と同じである．

表 5 は $|V|$，σ'，n のいずれかを変化させたときの実験結果である．表中の “-”は，メモリ不足のため計算が完了しなかったことを示す．また，頂点 ID 数 $|V|$ を変化させるときの頂点 ID は 182 個からランダムに選ぶ．そして，外部状態数 n について，$n = 4$ のときは，月曜日から木曜日までのメールのやり取りのデータを，$n = 5$ のときは，月曜日から金曜日までのメールのやり取りのデータを，$n = 6$ のときは月曜日から土曜日までのメールのやり取りのデータを，$n = 7$ のときは月曜日から日曜日までのメールのやり取りのデータをそれぞれ示している．人工データによる実験結果と同様に，どのパラメータの値を変化させても大幅に計算時間を短縮することが可能となった．また，列挙した FTS と rFTS の数の違いについても同様のことが言える．

さらに，個別のパラメータの変化における実験結果の考察を行う．まず，人の数を示すパラメータ $|V|$ を変化させたときの実験結果から，GTRACE-RS は変換系列の頂点 ID 数が最大でも 5 分に満たない時間で計算を終えることができることが分かった．一方，GTRACE は頂点 ID 数を増やすとメモリ不足で計算が完了せず，GTRACE-RS の方が大きなグラフ系列に適用できることが分かった．次に，最小支持度 σ' の値を小さくしていくと，GTRACE-RS の計算時間は指数関数的に増加しており，この結果は従来の頻出パターンマイニングの実験結果と同様である．最後に外部状態数 n を変化させたときの実験結果から，GTRACE は $n = 5$ 以上の場合メモリの容量が足りずに計算ができなかった．一方，GTRACE-RS は $n = 7$ のときでさえ，GTRACE の $n = 4$ のときの計算時間よりも短いという結果を得た．

人工データ，実データによる比較実験の結果，GTRACE と比べて GTRACE-RS の方が計算コスト，空間コスト共に優れていることが分かった．この結果が導かれるのは，マイニングされた FTS の数の結果から見ても分かるように，逆探索法によって探索空間の枝刈りをしたことに大きく起因する．また，

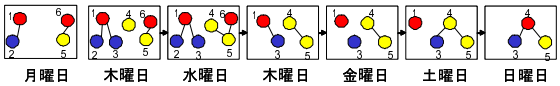

図 8　エンロン社電子メールデータをグラフ系列に変換した例

GTRACE-RS の方がどのパラメータの変化にも強いことから，GTRACE-RS の頑健性が高いことも予想される．

5. 関 連 研 究

本章では関連研究について紹介し，提案手法との関係について述べる．まず，GTRACE を改良した手法に GTRACE2 [10] と GTRACE(Clo-Span) [11] がある．GTRACE2 は和グラフにラベルを付与し，PrefixSpan の入力となる変換系列の数を減らすことによって計算効率を改善する手法であり，ラベルを変更する変換規則が少ない場合に特に有効である．GTRACE-RS と比較することが望まれるが，GTRACE2 がマイニングする FTS は rFTS でないものも含まれていることから，GTRACE-RS の方が効率よく探索できると考えられる．一方，GTRACE(CloSpan) は GTRACE の内部で呼び出される PrefixSpan を CloSpan [11] という手法に置き換えたものである．PrefixSpan はアイテム集合の系列から頻出な部分系列を列挙する手法であるが，CloSpan は頻出かつ飽和な部分系列を列挙する手法であるため，探索空間を枝刈りすることができ，計算効率の改善を可能とした手法である．飽和集合の詳しい定義は文献 [11] を参照されたい．この手法は GTRACE-RS に拡張することができるので，GTRACE-RS の計算時間をさらに短縮できることが考えられる．

さらに，GTRACE やその改良手法以外のものとして，文献 [6], [8], [9] がある．文献 [6] で対象としているグラフ系列は，系列中の各グラフの頂点数が固定され，その頂点間を結ぶ辺の数のみが増減する dynamic graph である．そして，dynamic graph から頻出パターンをマイニングする手法を提案している．一方，文献 [8] が対象とするグラフ系列は，頂点と辺が徐々に増える (減ることが許されない)evolving graph であり，evolving graph から頻出パターンをマイニングする手法を提案している．dynamic graph や evolving graph に比べ，提案手法を含めた GTRACE が対象とするグラフ系列はより汎用的なデータ構造である．そのため，dynamic graph や evolving graph に GTRACE や GTRACE-RS のアルゴリズムは適用できるが，文献 [6], [8] の提案手法を本研究で対象とするグラフ系列には適用できない．従って，GTRACE や GTRACE を改良したアルゴリズムの方が適用範囲が広い．

一方，FRISSMiner [9] はグラフ系列を対象として頻出パターンをマイニングする手法であるが，GTRACE と仮定が異なる．GTRACE ではグラフ系列はグラフが徐々に変化することを仮定しているが，FRISSMiner は必ずしもグラフが徐々に変化するという仮定を置く必要が無い．GTRACE が対象とするデータは，主に短い時間間隔でグラフの変化をサンプリングしたものであるが，FRISSMiner が対象とするデータは，細かい時間間隔でデータをサンプリングできない (グラフの変化がサンプリング周期に比べて非常に早い) ものを考えている．

6. ま と め

本稿では，逆探索の原理に基づき，探索空間の親子関係を表す 3 つの関数を定義することで，グラフ系列から直接 rFTS をマイニングする手法である GTRACE-RS を提案した．さらに，実装した本手法に関して人工データ，実データを用いて評価実験を行い，その有効性を検証した．

最後に本研究の今後の展望について述べる．本論文の評価実験では GTRACE との比較を行うのみだったが，前述の関連研究についても同様の比較実験を行うことが望まれる．また，GTRACE(CloSpan) の手法 [11] は GTRACE-RS に拡張することが可能なので，その実装を行いより効率のよいアルゴリズムを構築することが望まれる．一方，グラフ系列マイニングを使った実データの解析はあまり行われていない．今後，実際の応用の中で実データから有用な知識を発見する試みが行われることが望まれる．

文 献

[1] Alberto Sanfeliu and King-Sun Fu. A Distance Measure Between Attributed Relational Graphs for Pattern Recognition, *IEEE Transactions on Systems, Man and Cybernetic*, Vol. 13, pp. 353–362. (1983)

[2] David Avis and Komei Fukuda. Reverse Search for Enumeration. Discrete Applied Mathematics, Vol. 65 (1–3), pp. 21–46, (1996).

[3] Wai-Kai Chen. Graph Theory and Its Engineering Applications. Advanced Series in Electrical and Computer Engineering, Vol. 5 , pp. 19, (1997).

[4] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth, *Proc. of IEEE International Conference on Data Engineering* (ICDE), pp. 2–6. (2001)

[5] Xifeng Yan and Jiawei Han. gSpan: Graph-Based Substructure Pattern Mining. *Proc. of IEEE International Conference on Data Mining* (ICDM), pp. 721–724. (2002)

[6] Karsten M. Borgwardt, Hans-Peter Kriegel, and Peter Wackersreuther. Pattern Mining in Frequent Dynamic Subgraphs. *Proc. of IEEE International Conference on Data Mining* (ICDM), pp. 818–822. (2006)

[7] Akihiro Inokuchi and Takashi Washio. A Fast Method to Mine Frequent Subsequences from Graph Sequence Data. *Proc. of IEEE International Conference on Data Mining* (ICDM), pp. 303-312. (2008)

[8] Michele Berlingerio, Francesco Bonchi, Bjorn Bringmann, and Aristides Gionis. Mining Graph Evolution Rules. *Proc. of European Conference on Principles and Practice of Knowledge Discovery in Databases* (PKDD), pp. 115–130. (2009)

[9] Akihiro Inokuchi and Takashi Washio. Mining Frequent Graph Sequence Patterns Induced by Vertices. *Proc. of SIAM International Conference on Data Mining* (SDM), pp. 466-477. (2010)

[10] Akihiro Inokuchi, Takashi Washio. GTRACE2:Improving Performance Using Labeled Union Graphs. *Proc. of Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD)* , pp. 177–188, (2010)

[11] 岸本卓也，猪口明博，鷲尾隆. 飽和系列パターンマイニングを用いたグラフ系列マイニングの高速化. 人工知能学会全国大会 , 3A2-4, (2010)

[12] Enron Email Dataset, `http://www.cs.cmu.edu/~enron/`