情報処理学会論文誌　データベース　Vol. 1　No. 3　20–35 (Dec. 2008)

拡張出現マッチングを用いた制約付きノイズ許容極小順序木パターンの発見

尾崎 知伸[†1]　大川 剛直[†2]

近年，大量のパターンが抽出されるという構造データを対象とした頻出パターン発見の問題に対し，(1) 頻出パターンの代表元のみを発見する手法や，(2) 利用者により与えられる制約を満たすパターンのみを発見する手法などが提案されている．本論文では，順序木データベースを対象とした両者の統合アプローチとして，制約下でのノイズを許容した極小元である，(1) 制約付き δ-フリー順序木パターンと，(2) 制約付き Δ-トレランス順序木パターンの発見問題について議論する．この問題を解決するために，本論文では，拡張出現マッチングと，それに基づく枝刈り手法をともなう 3 種のアルゴリズムを提案する．合成データと実データを用いた比較実験により，抽出されるパターン数や実行時間の観点から，提案手法の有効性が確認された．

Mining Noise-tolerant Minimal Constrained Ordered Subtrees by Using Extended Occurrence Matching

Tomonobu Ozaki[†1] and Takenao Ohkawa[†2]

Frequent pattern miners for structured data often discover huge number of patterns. To alleviate this problem, two major approaches, (1) condensed representation mining and (2) constraint-based mining, have been proposed. In this paper, as a technique for integrating these two approaches in ordered subtree mining, we focus on mining 'noise-tolerant minimal patterns under constraints' and discuss the problems of mining (1) δ-free constrained ordered subtrees and (2) Δ-tolerance constrained ordered subtrees. To achieve this objective, we propose three kinds of algorithms having pruning capability based on extended occurrence-matching. The results of experiments with synthetic and real world datasets show that, compared with a naive algorithm, the proposed algorithms succeed in reducing the number of extracted patterns and execution time.

†1 神戸大学自然科学系先端融合研究環
Organization of Advanced Science and Technology, Kobe University
†2 神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University

1. はじめに

近年，頻出パターン発見を中心に，構造データを対象としたマイニング手法の研究が活発に行われている[6),14)]．構造データを対象とした頻出パターン発見では，大量のパターンが抽出されるという問題が指摘されているが，この問題に対し，(1) 代表元のみを抽出することで頻出パターン集合の圧縮表現を求める手法[7),11),15)] や，(2) 利用者が制約を与えることで積極的に抽出すべきパターンを限定する手法[13),17)]，(3) 制約下での代表元（極大元）を求める統合アプローチ[12)] などが提案されている．特に統合アプローチは，制約を用いて能動的にパターンを限定し，そのうえで圧縮表現を求めるものであり，目的の異なる両アプローチを組み合わせることで，より効果的なパターン発見が実現されることが期待できる．しかし，これまでに提案された統合アプローチ[12)] は，逆単調制約（制約の種類については後述する）のみを対象としており，必ずしも任意の制約を扱えるわけではない．そこで本論文では，より柔軟なパターン発見を実現するため，逆単調制約に加え，単調制約をも対象とした統合アプローチについて議論する．

一方，代表元を求める手法の問題点として，ノイズに対する脆弱性が指摘されている[4),5)]．加えて，得られたパターンをその後の分析に利用することを考えた場合，MDL 原理などの観点から（極大元ではなく）極小元を用いる方が適切であるという報告もなされている[3),9)]．これらのことから，構造データを対象とした頻出パターン発見において，ノイズを考慮した極小元の発見は 1 つの重要な課題であると考えられる．しかし筆者の知る限り，構造データを対象としたノイズを考慮した代表元の発見，および代表元としての極小元の発見に関して，ほとんど研究が行われていない．

これらのことを背景に，本論文では，順序木データベースを対象とした統合アプローチの 1 つとして，単調・逆単調制約下でのノイズを考慮した極小元，すなわち**制約付きノイズ許容極小順序木パターン**の発見について議論する．具体的には，(1) 支持度の差に着目した**制約付き δ-フリー順序木パターン**（以後 δ-FCost）と (2) 支持度の比に着目した**制約付き Δ-トレランス順序木パターン**（以後 Δ-TCost）[⋆1]の発見について議論する．δ-FCost および Δ-TCost は，利用者が能動的に与える制約を満たす頻出パターンのうち，軽微な支持度の違いを考慮したうえでの極小元である．したがって，δ-FCost の集合を $\mathcal{F}_\delta$，Δ-TCost の

⋆1 これらの名称は，頻出アイテム集合発見における支持度の差に着目した代表元 δ-フリーパターン[2)] と支持度の比に着目した代表元 δ-トレランス飽和パターン[5)] に由来する．

集合を $\mathcal{F}_\Delta$ とした場合，頻出パターン集合 $\mathcal{F}$ とそのうち制約を満たすパターンの集合 $\mathcal{F}_C$ に対し，$\mathcal{F}_\delta, \mathcal{F}_\Delta \subseteq \mathcal{F}_C \subseteq \mathcal{F}$ が成り立つ．ところで，類似するパターンは類似する支持度を持つことも多いが，δ-FCost や Δ-TCost の発見では，パターンの形状と支持度の類似性の両面から頻出パターン集合の圧縮を試みていると考えることができる．また，極小性と単調制約を同時に考慮することで，過度に単純化（一般化）されたパターンの導出を回避することも達成されると考えられる．

δ-FCost や Δ-TCost におけるこれらの特徴は，たとえば，Web ページの閲覧履歴を木構造で表現した Web アクセスログ木[16] などを対象とした分析において有効であると考えられる．Web ページの閲覧に関しては，たとえば，「トップページから入り，メニューをたどり……」など，ある程度のパターンを事前に想定することができる．これに対し，たとえば，単調制約を用いて大きさの小さなパターンや分岐の少ないパターンなどを排除することで，事前に想定できる典型的なパターンの発見を回避し，そこから派生したようなパターンを中心としたパターン発見が実現されると考えられる．一方，分析時にノイズを考慮することで，Web ページ閲覧者の操作ミスによって生じるノイズに対し，ある程度対処できると考えられる．

δ-FCost および Δ-TCost の効率的な発見を実現するために，本論文では，まず，**制約付き δ-出現マッチング**を提案するとともに，これを探索戦略の異なる既存の頻出順序木発見手法[1),8),11)] に組み込むことで，3 種の δ-FCost 発見アルゴリズムを提案する．さらに，δ-出現マッチングと Δ-TCost の関係に着目し，δ-FCost 発見アルゴリズムの一部を改変することで，Δ-TCost の発見を実現する．

以下に本論文の構成を示す．2 章で用語の導入を行い，対象とする問題の形式化を行う．3 章で制約付き δ-出現マッチングを導入し，ついで 4 章で δ-FCost および Δ-TCost 発見のための 3 種の手法を提案する．5 章で実験結果を示し，最後に 6 章でまとめを行う．

2. 準　　備

本章では，文献 1)，2)，6)，8)，11) などに従い用語を導入するとともに，本論文で対象とするデータマイニング問題の形式的な定義を与える．

ラベル集合 $\mathcal{L}$ 上の**順序木**とは，五項組 $t = (V, E, S, r, L)$ で表される r を根とする木である．ここで，V は節点の集合を，辺集合 $E \subseteq V \times V$ は節点間の親子関係を，$S \subseteq V \times V$ は節点間の兄弟関係をそれぞれ表す．また $L : V \to \mathcal{L}$ はラベル関数である．ここで，$\mathcal{L}$ 上の順序木の全体集合を $\mathcal{T}_\mathcal{L}$ と表記する．また簡略化のため，以後，順序木を単に木と呼ぶ．

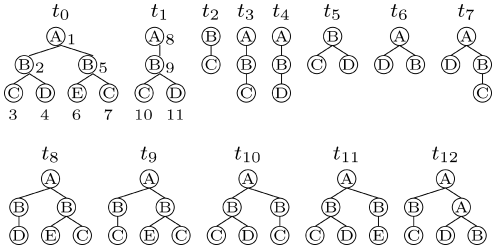

図 1　順序木の例
Fig. 1　Example of ordered trees.

木 $s = (V_s, E_s, S_s, r_s, L_s)$ と $t = (V, E, S, r, L)$ に対し，(1) $(v_1, v_2) \in E_s \Leftrightarrow (\phi(v_1), \phi(v_2)) \in E$，(2) $(v_1, v_2) \in S_s \Leftrightarrow (\phi(v_1), \phi(v_2)) \in S$，(3) $L_s(v) = L(\phi(v))$ を満たす単射 $\phi : V_s \to V$ が存在するとき，s を t の**部分木**と定義し，$s \prec t$ と記す．上記の条件を満たす ϕ により得られる t 中の頂点集合 $\{\phi(v) \in V \mid v \in V_s\}$ を，t における s の**出現**と定義する．また，t における s の出現の全集合を Φ_t^s と表記する．**図 1** に順序木の例を示す．図中において，たとえば $t_3 \prec t_0$，$t_2 \prec t_1$ であり，$\Phi_{t_0}^{t_3} = \{\{Ⓐ_1, Ⓑ_2, Ⓒ_3\}, \{Ⓐ_1, Ⓑ_5, Ⓒ_7\}\}$，$\Phi_{t_1}^{t_2} = \{\{Ⓑ_9, Ⓒ_{10}\}\}$ である．

木 $t = (V, E, S, r, L)$ において，節点 $v \in V$ のラベルを $\mathrm{l}(v)$ と表記する．また v の深さ $\mathrm{d}(v)$ を根 r から v への経路上の辺の数と定義する．t の高さを $\mathrm{h}(t) = \max(\{\mathrm{d}(v) \mid v \in V\})$，大きさを $|t| = |V|$ と定義する．たとえば図 1 中の t_0 において，$\mathrm{d}(Ⓓ_4) = 2$，$\mathrm{l}(Ⓓ_4) = D$，$\mathrm{h}(t_0) = 2$，$|t_0| = 7$ である．

ちょうど 1 つの葉を持つ木を**直列木**と呼ぶ．また木 t が直列木であるとき，$\mathrm{path}(t)$ と表記する．木 t の最も右側の葉を**最右葉**と呼び，$\mathrm{rl}(t)$ と表記する．同様に，最も左側の葉を**最左葉**と呼び，$\mathrm{ll}(t)$ と表記する．t よりちょうど 1 段階一般的な木の集合を $\mathrm{P}(t) = \{s \prec t \mid |s| + 1 = |t|\}$ と定義し，各 $s \in \mathrm{P}(t)$ を t の**親**と呼ぶ．t から節点 v を取り除いた木を t/v と表記する．特に，t から $\mathrm{ll}(t)$ を取り除いた木を $\mathrm{p_r}(t) = t/\mathrm{ll}(t)$，同じく $\mathrm{rl}(t)$ を取り除いた木を $\mathrm{p_l}(t) = t/\mathrm{rl}(t)$ とそれぞれ表記する．一方，t が 3 以上の葉を持つ場合，t から $\mathrm{rl}(t)$，$\mathrm{ll}(t)$ 以外の葉を 1 つ取り除いた木を $\mathrm{p_c}(t)$ と表記する．また t が 4 以上の葉を持つ場合，$\mathrm{p_c}(t)$ のうち，$\mathrm{p_l}(\mathrm{p_c}(t)) = \mathrm{p_l}(\mathrm{p_l}(t))$ が成り立つ，すなわち右から 2 番目の葉を取り除いた木を $\mathrm{p_c^l}(t)$，それ以外を $\mathrm{p_c^r}(t)$ と表記し，必要に応じて区別する．なお，葉の数が 2 以下の木 t に対しては，$\mathrm{p_c}(t)$ は存在しないものとする．同様に，3 以下の木 t に対しては，$\mathrm{p_c^r}(t)$ は存在しないものとする．図 1 において，t_2，t_3，t_4 は直列木である．$\mathrm{rl}(t_0) = Ⓒ_7$，$\mathrm{ll}(t_1) = Ⓒ_{10}$ であり，ま

た $\mathrm{p_l}(t_0) = t_{11}$，$\mathrm{p_c^l}(t_0) = t_{10}$，$\mathrm{p_c^r}(t_0) = t_9$，$\mathrm{p_r}(t_0) = t_8$，$\mathrm{P}(t_0) = \{t_8, t_9, t_{10}, t_{11}\}$ である．

木 t に，最左葉として節点 v を追加することで得られる木を $v \bullet t$ と表記する．同様に，t に最右葉として節点 v を追加することで得られる木を $t \cdot v$ と表記する．なお追加する節点 v を，$(\mathrm{d}(v), \mathrm{l}(v))$ とも表記する．一方，t 中の節点 v に対し，v の親，子または直接の兄弟として節点を加えることで得られる木の集合を $\mathrm{B}(t, v)$ とする．木 t と $t^* \succ t$ および t 中の節点 v に対し，$\forall t_b \in \mathrm{B}(t, v)$（$t_b \not\prec t^*$）が成り立つとき，すなわち，$t$ に対し v の親，子または直接の兄弟以外の位置に繰り返し節点を加えることで t^* が得られる場合，$t \overset{-v}{\prec} t^*$ と表記する．たとえば図 1 において，$t_1 = (2, C) \bullet t_4 = t_3 \cdot (2, D)$ である．また，t_2 の Ⓑ の末子，すなわち Ⓒ の直接の弟として節点 Ⓓ を追加することで t_5 が得られるので，$t_5 \in \mathrm{B}(t_2, \text{Ⓑ})$ かつ $t_5 \in \mathrm{B}(t_2, \text{Ⓒ})$ である．同様に，$t_1 \in \mathrm{B}(t_5, B)$，$t_1 \notin \mathrm{B}(t_5, C)$ である．一方，t_1 と t_0 の間には，$t_1 \overset{-D}{\prec} t_0$，$\neg(t_1 \overset{-A}{\prec} t_0)$ が成り立つ．

順序木 t が制約 C を満たすとき，$\mathrm{C}(t)$ と書く．t が制約 C_A を満たすとき，その任意の部分木 s もその制約を満たす場合，すなわち $\forall s \prec t\ (\mathrm{C}_A(t) \to \mathrm{C}_A(s))$ が成り立つとき，C_A を**逆単調制約**と定義する．たとえば，$s \prec t$ なる 2 つの木の大きさに関して $|s| \leq |t|$ が成り立つ．したがって，「木 t の大きさは k 以下でなければならない」という制約 $\mathrm{C}_A(t) \equiv |t| \leq k$ は，逆単調制約となる．同様に，$\forall t \prec t'(\mathrm{C}_M(t) \to \mathrm{C}_M(t'))$ が成り立つとき，C_M を**単調制約**と定義する．たとえば，$t \prec t'$ なる木の高さに関しては $\mathrm{h}(t) \leq \mathrm{h}(t')$ が成り立つので，「木 t の高さは h 以上でなければならない」という制約 $\mathrm{C}_M(t) \equiv \mathrm{h}(t) \geq h$ は，単調制約となる．

N 個の順序木からなる集合 $D = \{t_1, t_2, \cdots, t_N\}$ を順序木データベースと呼ぶ．関数 $\mathrm{Oc}(t, d)$ を，$t \prec d$ なら 1，そうでないなら 0 を返す関数とし，D における木 t の**支持度**を $sup(D, t) = \sum_{d \in D} \mathrm{Oc}(t, d)$ と定義する．また便宜上，大きさ 0 の順序木を $\perp$ と表記し，$\forall t \in \mathcal{T}_{\mathcal{L}}(\perp \prec t)$，$sup(D, \perp) = |D|$ と定義する．

D と最小支持度 $\sigma \geq 1$ に対し，$sup(D, t) \geq \sigma$ なる木を**頻出順序木**と定義する．単調・逆単調制約 $\mathrm{C} = \mathrm{C}_M \cup \mathrm{C}_A$ および**許容ノイズ数**（支持度の誤差）を表す非負の整数 $\delta \geq 0$，$\sigma > \delta$，D が与えられたとき，支持度差 δ 以内の制約 C を満たす頻出順序木集合における極小元，すなわち

$$sup(D, t) \geq \sigma \wedge \mathrm{C}(t) \wedge \nexists s \prec t\ (s \neq t \wedge \mathrm{C}(s) \wedge sup(D, s) - sup(D, t) \leq \delta)$$

を満たす順序木 t を，**制約付き δ-フリー順序木パターン**（δ-FCost）と定義する．δ-FCost は，同一頻度を持つパターン集合を同値類とした場合の極小パターンを，支持度差（許容ノイズ数）と制約を考慮し，拡張したものである．すなわち，制約 C を満たす頻出順序木 t に対する同値類を，t との支持度差 δ 以内の C を満たす順序木の集合と考え，その集合の部分木の関係 $\prec$ に基づく極小元の 1 つが t と一致する場合，t を δ-FCost と定義する．

一方，データベース中に一様にノイズが存在する場合など，同一の基準 δ ではなく，パターンごとにその支持度の大きさに応じたノイズ許容基準を用いる方が適切である場合も考えられる．この問題に対処するため，順序木 t に対する制約 C と**許容ノイズ率** $\Delta \geq 1$ に基づく同値類を，t との支持度比が Δ 以内の C を満たす順序木集合と定義し，その極小元を抽出することを考える．形式的には，許容ノイズ率を表す実数 $\Delta \geq 1$ および $\mathrm{C} = \mathrm{C}_M \cup \mathrm{C}_A$，$D$，$\sigma \geq 1$ が与えられたとき，支持度 σ 以上を前提に，支持度比 Δ 以内の制約 C を満たす順序木集合における極小元，すなわち，

$$sup(D, t) \geq \sigma \wedge \mathrm{C}(t) \wedge \nexists s \prec t\ (\ s \neq t \wedge \mathrm{C}(s) \wedge sup(D, s)/sup(D, t) \leq \Delta)$$

を満たす順序木 t を**制約付き Δ-トレランス順序木パターン**（Δ-TCost）と定義する．

本論文では，データベース D および単調・逆単調制約 $\mathrm{C} = \mathrm{C}_M \cup \mathrm{C}_A$，許容ノイズ数 $\delta \geq 0$，最小支持度 $\sigma > \delta$ を入力とし，D 中のすべての δ-FCost を発見する問題，すなわち**制約付き δ フリー順序木パターン発見問題**について議論する．また同様に，D，C，σ と許容ノイズ率 $\Delta \geq 1$ を入力とし，D 中のすべての Δ-TCost を発見する問題，すなわち**制約付き Δ-トレランス順序木パターン発見問題**についても議論を行う．

3. 出現マッチングの拡張

本章では，まず δ-FCost および Δ-TCost の非逆単調性を示す．その後，枝刈り基準の基礎として，飽和順序木発見の枝刈り基準として用いられる出現マッチング[7),11)]を，許容ノイズ数および制約の観点から拡張した**制約付き δ-出現マッチング**を提案する．

3.1 δ-FCost および Δ-TCost の非逆単調性

単調・逆単調制約 $\mathrm{C} = \mathrm{C}_M \cup \mathrm{C}_A$，データベース D，許容ノイズ数 δ，許容ノイズ率 Δ および頻出順序木 t に対し，

$$\mathrm{dM}_D^{\delta,\mathrm{C}}(t) = \{s \in \mathrm{P}(t) \mid \mathrm{C}_M(s),\ sup(D, s) - sup(D, t) \leq \delta\}$$
$$\mathrm{rM}_D^{\Delta,\mathrm{C}}(t) = \{s \in \mathrm{P}(t) \mid \mathrm{C}_M(s),\ sup(D, s)/sup(D, t) \leq \Delta\}$$

と定義する．すなわち $\mathrm{dM}_D^{\delta,\mathrm{C}}(t)$ は，t との支持度差が δ 以内の制約 C_M を満たす t の部分木の集合である．また同様に $\mathrm{rM}_D^{\Delta,\mathrm{C}}(t)$ は，t との支持度比が Δ 以内の制約 C_M を満たす t の部分木の集合である．このとき，制約 C を満たす $\mathrm{dM}_D^{\delta,\mathrm{C}}(t) = \emptyset$（$\mathrm{rM}_D^{\Delta,\mathrm{C}}(t) = \emptyset$）なる頻

出順序木 t は δ-FCost（Δ-TCost）である．δ-FCost および Δ-TCost 発見の最も単純な方法は，既存の頻出順序木列挙アルゴリズム[1),8)] を用いてすべての頻出順序木を列挙し，後処理として各木 t が制約 C を満たし，かつ $\mathrm{dM}_D^{\delta,\mathrm{C}}(t)=\emptyset$ または $\mathrm{rM}_D^{\Delta,\mathrm{C}}(t)=\emptyset$ であるかを確認すればよい．しかし，より効率的なパターン発見の実現には，後処理としての確認による方法ではなく，枝刈りをともなう探索を用いることが望ましい．

アイテム集合データベースを対象とした場合は，δ-フリーパターンの逆単調性[2)]，すなわち「δ-フリーでないパターンの上位集合は δ-フリーパターンとはなりえない」という性質に基づく枝刈りが利用できる．一方，δ-FCost や Δ-TCost に関しては，この逆単調性が成り立たない．たとえば図 1 において，$D=\{t_7,t_{12}\}$，$\sigma=1$，$\delta=0$ かつ制約なし（$\mathrm{C}=\emptyset$）とすると，$sup(D,\perp)=sup(D,t_3)=sup(D,t_6)=2$，$sup(D,t_7)=1$ より，t_6 は δ-FCost ではないが，t_7（$\succ t_6$）は δ-FCost である．また同様に $\Delta=1$ としたとき，t_6 は Δ-TCost ではないが，t_7 は Δ-TCost である．したがって，効率的なパターン発見の実現には，別の基準での枝刈りが必要となる．本論文では，δ-FCost および Δ-TCost 発見における枝刈り基準の基礎として，**制約付き δ-出現マッチング**を提案する．

3.2　制約付き δ-出現マッチング

制約付き δ-出現マッチングは，飽和順序木発見において提案された出現マッチング[6)] を，制約およびノイズを考慮するように拡張したものである．

3 つの木 t，$s\in\mathrm{P}(t)$ および $d\succ s$ に対し，d における s の全出現が d における t の出現に覆われる，すなわち $\forall V_s\in\Phi_d^s\,\exists V_t\in\Phi_d^t$（$V_s\subset V_t$）であるとき，**$s$ と t は d に関して等しい**といい，$s\overset{d}{=}t$ と表記する．たとえば，図 1 中の t_0，t_2，t_3，t_5（$t_2\in\mathrm{P}(t_3)$，$t_2\in\mathrm{P}(t_5)$，$t_2\prec t_0$，$t_3\prec t_0$，$t_5\prec t_0$）において，

$$\Phi_{t_0}^{t_2}=\{\{\textcircled{B}_2,\textcircled{C}_3\},\{\textcircled{B}_5,\textcircled{C}_7\}\}$$
$$\Phi_{t_0}^{t_3}=\{\{\textcircled{A}_1,\textcircled{B}_2,\textcircled{C}_3\},\{\textcircled{A}_1,\textcircled{B}_5,\textcircled{C}_7\}\}$$
$$\Phi_{t_0}^{t_5}=\{\{\textcircled{B}_2,\textcircled{C}_3,\textcircled{D}_4\}\}$$

であるので，$t_2\overset{t_0}{=}t_3$ であるが，$t_2\overset{t_0}{=}t_5$ ではない．

単調制約 C_M，データベース D，許容ノイズ数 δ と，$s=t/v$ なる 2 つの木 s，t に対し，$\mathrm{C}_M(s)$ かつ

$$sup(D,s)-\left|\{d\in D\mid s\overset{d}{=}t\}\right|\quad\leq\quad\delta$$

が成り立つとき，**s と t は D および C_M に関して制約付き δ-出現マッチする**といい，

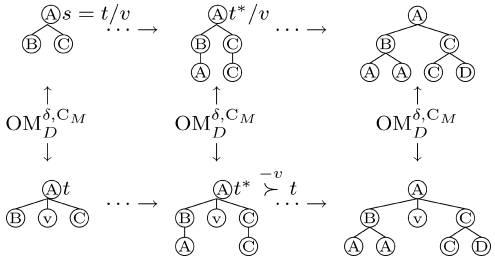

図 2　δ 出現マッチングに関する順序木間の関係
Fig. 2　Relationship among subtrees on δ-occurrence matching.

$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(s,t)$ と表記する．$sup(D,t)\geq\left|\{d\in D\mid s\overset{d}{=}t\}\right|$ より，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(s,t)$ であれば $sup(D,s)-sup(D,t)\leq\delta$ が成り立つ．これにより，支持度差 δ 以内の $s\prec t$（$s\in\mathrm{P}(t)$）が存在するため，t は δ-FCost とはなりえない．

たとえば図 1 において，$D=\{t_1,t_{10},t_{11}\}$ および $\delta=1$，$\mathrm{C}=\emptyset$ としたとき，$sup(D,t_2)=3$ および $t_2\overset{t_1}{=}t_5$，$t_2\overset{t_{11}}{=}t_5$ より

$$sup(D,t_2)-|\{d\in D\mid t_2\overset{d}{=}t_5\}|=3-2\leq 1$$

が得られ，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(t_2,t_5)$ が成り立つ．またこれにより，t_5 は δ-FCost とはなりえない．

以下に，制約付き δ-出現マッチングに関する補題を示す（**図 2** 参照）．

補題 1　単調制約 C_M，データベース D，許容ノイズ数 δ と木 $s=t/v$ に対し，s と t が制約付き δ-出現マッチする（$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(s,t)$）ならば，$t$ に対し v の親，子または直接の兄弟以外の位置に繰り返し節点を加えることで得られる任意の木 $t^*\overset{-v}{\succ}t$ は，必ず t^*/v と制約付き δ-出現マッチする（$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(t^*/v,t^*)$）．すなわち，以下の関係が成り立つ．

$$\forall t^*\overset{-v}{\succ}t(\ \mathrm{OM}_D^{\delta,\mathrm{C}_M}(s,t)\rightarrow\mathrm{OM}_D^{\delta,\mathrm{C}_M}(t^*/v,t^*)\ )$$

証明　3 つの集合 $D_1^p=\{d\in D\mid s\overset{d}{=}t\}$，$D^n=D\setminus D_1^p$，$D_2^p=\{d\in D\mid t^*/v\overset{d}{=}t^*\}$ を考える．このとき $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(t^*/v,t^*)$ が成り立つためには，

$$sup(D,t^*/v)-\left|\{d\in D\mid t^*/v\overset{d}{=}t^*\}\right|=sup(D^n,t^*/v)+sup(D_1^p,t^*/v)-|D_2^p|\leq\delta\qquad(1)$$

および $\mathrm{C}_M(t^*/v)$ を示せばよい．

$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(s,t)$ と $s \prec t^*/v$ より

$$\begin{aligned} sup(D,s) - |D_1^p| &= sup(D^n,s) + sup(D_1^p,s) - |D_1^p| \\ &= sup(D^n,s) + |D_1^p| - |D_1^p| \\ &= sup(D^n,s) \ \leq \ \delta \end{aligned}$$

$$sup(D^n, t^*/v) \leq sup(D^n, s) \ \leq \ \delta \tag{2}$$

定義より，$s \stackrel{d}{=} t$ なる順序木 $s = t/v$，t および d に対し $\forall t^* \stackrel{-v}{\succ} t (t^* \prec d \rightarrow t^*/v \stackrel{d}{=} t^*)$ が成り立つので

$$sup(D_1^p, t^*/v) = |D_2^p| \tag{3}$$

が得られる．これにより，

$$\begin{aligned} &sup(D^n, t^*/v) + sup(D_1^p, t^*/v) - |D_2^p| \ \text{(式 (1))} \\ &\quad = sup(D^n, t^*/v) + |D_2^p| - |D_2^p| \ \ \text{(式 (3) より)} \\ &\quad = sup(D^n, t^*/v) \leq \delta \ \ \text{(式 (2) より)} \end{aligned}$$

一方，単調制約の定義と，$\mathrm{C}_M(s)$ および $s \prec t^*/v$ より，$\mathrm{C}_M(t^*/v)$ が成り立つ． □

補題 1 により，$s = t/v$ に対し $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(s,t)$ が成り立てば，$t \stackrel{-v}{\prec} t^*$ なる木 t^* は，$t^*/v \in \mathrm{P}(t^*)$ と制約付き δ-出現マッチするので δ-FCost とはなりえない．したがって，たとえば文献 1)，8)，11) など，一般から特殊の方向へと探索を行う順序木列挙アルゴリズム $\mathcal{A}$ を前提とした場合，$\mathcal{A}$ において t とそこから生成される $t^* \stackrel{-v}{\succ} t$ を探索の対象から外すことが，それ以外の木の生成に影響を与えなければ，t を枝刈りすることが可能である．このことは，補題 1 に基づく枝刈りが，採用するアルゴリズム $\mathcal{A}$ と v の位置に強く依存することを意味している．

3.3　Δ-TCost と制約付き δ-出現マッチング

本節では，制約付き δ-出現マッチングと Δ-TCost との間の関係について考察し，Δ-TCost 発見における枝刈り手法導入の基礎を与える．

制約付き δ-出現マッチングと Δ-TCost の間には，以下の関係が成り立つ．

補題 2　木 $s = t/v$ およびデータベース D，最小支持度 σ，許容ノイズ率 Δ，許容ノイズ数 $\delta = \lfloor(\Delta - 1)\sigma\rfloor$，単調制約 C_M に対し，以下の関係が成り立つ．

$$\forall t^* \stackrel{-v}{\succ} t \left(\mathrm{OM}_D^{\delta,\mathrm{C}_M}(s,t) \rightarrow \left(sup(D,t^*) \geq \sigma \rightarrow \frac{sup(D,t^*/v)}{sup(D,t^*)} \leq \Delta \right) \right)$$

証明　$A = \left| \{ d \in D \mid t^*/v \stackrel{d}{=} t^* \} \right|$，$B$，$B'$ を非負の整数とし，t^*/v と t^* の支持度を，$sup(D, t^*/v) = A + B + B'$，$sup(D, t^*) = A + B$ と表現する．このとき，

$$\frac{sup(D,t^*/v)}{sup(D,t^*)} = 1 + \frac{B'}{sup(D,t^*)} \tag{4}$$

である．一方，補題 1 より，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(t^*/v, t^*)$ であり

$$\begin{aligned} sup(D, t^*/v) - \left| \{ d \in D \mid t^*/v \stackrel{d}{=} t^* \} \right| = B + B' &\leq \delta = \lfloor(\Delta - 1)\sigma\rfloor \\ &\leq (\Delta - 1)\sigma \end{aligned}$$

が得られ，これより

$$1 + \frac{B + B'}{\sigma} \leq \Delta \tag{5}$$

式 (4)，(5) および $sup(D, t^*) \geq \sigma$ より

$$\frac{sup(D,t^*/v)}{sup(D,t^*)} = 1 + \frac{B'}{sup(D,t^*)} \leq \ 1 + \frac{B + B'}{\sigma} \leq \Delta$$

□

補題 2 は，許容ノイズ率 Δ に対し $\delta = \lfloor(\Delta - 1)\sigma\rfloor$ とした場合，もし $s = t/v$ と t が制約付き δ-出現マッチする（$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(s,t)$）ならば，$t$ に対し v の親，子または直接の兄弟以外の位置に繰り返し節点を加えることで得られる任意の頻出木 $t^* \stackrel{-v}{\succ} t$ と t^*/v との頻度の比は Δ 以下であり（$\frac{sup(D,t^*/v)}{sup(D,t^*)} \leq \Delta$），結果として，$t^*$ は Δ-TCost になりえないことを表している．したがって，$\delta = \lfloor(\Delta - 1)\sigma\rfloor$ とし，$\mathrm{dM}_D^{\delta,\mathrm{C}}(t)$ の代わりに $\mathrm{rM}_D^{\Delta,\mathrm{C}}(t)$ を用いることで，4 章で提案する δ-FCost 発見手法をそのまま Δ-TCost 発見に適用することが可能である．

4.　制約付き δ フリー順序木パターンの発見

これまでに，（制限付き）最右拡張[1),11)] を用いた，(1) 深さ優先探索[1),16)]，(2) 幅優先探索[8)]，(3) 深さ優先＋幅優先探索[11)] に基づく頻出順序木発見アルゴリズムがそれぞれ提案されている．これらのうち，どのアルゴリズムが優れているかは必ずしも自明ではなく，対象とする問題やデータ集合，動作環境などに合わせて適切にアルゴリズムを選択することが重要であると考えられる．本論文では，より広範囲な問題に柔軟に対応することを目的に，既存の各種頻出順序木発見アルゴリズムに補題 1 を基にした枝刈り手法をそれぞれ導入し，各探索戦略に基づく制約付き δ フリー順序木パターン発見アルゴリズムを構築する．

まず，既存アルゴリズムに対する枝刈り導入における基本的な考え方を示す．詳しくは後

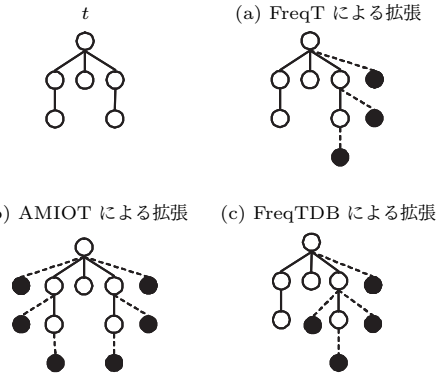

図 3　各アルゴリズムにおける木 t の拡張方法
Fig. 3　Extensions of a tree t.

述するが，**図 3** に，対象とする各頻出順序木発見アルゴリズムにおける木の拡張方法を図示する．各アルゴリズムは，木 t に対し，黒い円（●）で表される位置に節点を追加することで，新たな順序木を得る操作を繰り返す．したがって，黒い円に隣接しない，すなわち黒い円の親，子または直接の兄弟以外の節点 v を t から取り除いた木 s と，t 自身とが，制約付き δ-出現マッチするならば，木 t から生成される任意の木 t^* は，$t^* \overset{-v}{\succ} t$ を満たすので，補題 1 に基づく枝刈りを適用することが可能となる．以下，この基本的な考えを，各探索戦略に基づく頻出順序木発見アルゴリズムに適用する．

4.1　深さ優先探索による δ-FCost の発見

順序木 t に対し，その最右枝上に節点を加えることで新たな順序木集合を得る操作，すなわち

$$\mathrm{refine}(t) = \{t \cdot (d, l) \mid 1 \leq d \leq \mathrm{d}(\mathrm{rl}(t)) + 1, l \in \mathcal{L}\}$$

を得る操作を**最右拡張**[1),16)]と呼ぶ．たとえば図 1 において，$t_7 \in \mathrm{refine}(t_6)$，$t_0 \in \mathrm{refine}(t_{11})$，$t_0 \notin \mathrm{refine}(t_{10})$ である．また，1 つの節点 v ($\mathrm{d}(v) = 0, \mathrm{l}(v) \in \mathcal{L}$) からなる木に対し，最右拡張を繰り返し適用することで，すべての順序木を重複なく列挙することが可能である[1)]．**図 4** に，最右拡張による深さ優先探索を用いた頻出順序木発見アルゴリズム FreqT を示す．図中において $\mathcal{F}_1$ は，大きさ 1 の頻出順序木の全体集合 $\mathcal{F}_1 = \{t \in \mathcal{T}_\mathcal{L} \mid |t| = 1, sup(D, t) \geq \sigma\}$ を表す．

FreqT では，順序木 t の最右枝上に節点を加えることで，新たな順序木集合を得る．FreqT

Algorithm FreqT ($\mathcal{F}_1$, D, σ)
1: **for each** $t \in \mathcal{F}_1$
2:　FreqT-Enum(t, D, σ)
3: **end for**

Function FreqT-Enum(t, D, σ)
1: **output** t
2: **for each** d ($1 \leq d \leq \mathrm{d}(\mathrm{rl}(t)) + 1$)
3:　**for each** $l \in \mathcal{L}$
4:　　$s := t \cdot (d, l)$
5:　　**if** $sup(D, s) \geq \sigma$ **then**
6:　　FreqT-Enum(s, D, σ)
7:　**end for**
8: **end for**

図 4　頻出順序木発見アルゴリズム FreqT
Fig. 4　Pseudo code of FreqT.

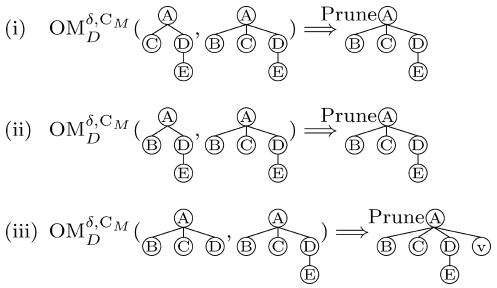

図 5　dFCost$^{\mathrm{DF}}$ における枝刈りの例
Fig. 5　Pruning in dFCost$^{\mathrm{DF}}$.

における t から t' への拡張を図 3 (a) に図示する．図中において，木 t に黒い円で表される節点のいずれかを追加することで，新たな順序木 t' が得られる．

効率の良い δ-FCost 発見アルゴリズムを実現するために，FreqT に対し，補題 1 に基づいた枝刈りを導入する（**図 5** 参照）．以下に示す補題 3 により，FreqT を用いた δ-FCost 発見において，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_r}(t), t)$ もしくは $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c}(t), t)$ なる木 t に対し，枝刈りを適

用することが可能である．すなわち木 t が，自身から最右葉以外の葉を取り除いた木と制約付き δ-出現マッチするならば，t の拡張をやめることが可能である．

補題 3（右＋中央木刈り） 単調制約 C_M，データベース D，許容ノイズ数 δ に対し，$\neg\mathrm{path}(t)$ なる木 t が，(1) $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_r}(t),t)$，$\mathrm{p_r}(t)=t/v$（図 5 の (i)），または，(2) $\exists \mathrm{p_c}(t)=t/v$ *s.t.* $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c}(t),t)$（図 5 の (ii)）を満たすとき，$t$ を枝刈りすることは，δ-FCost の列挙に影響を与えない（安全である）．

証明 $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_r}(t),t)$ の場合を考える．FreqT は，t に対し，その最右枝上に節点を追加することで拡張を行う．また $v=\mathrm{ll}(t)$ より，v は t の最右枝上には存在しないので，t から得られる木 t^* は，必ず $t^* \succ^{-v} t$ を満たす．したがって，t の枝刈りは，$t^* \succ^{-v} t$ なる木 t^* 以外の木の生成に影響を与えない．一方，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_r}(t),t)$ と補題 1 より，t^* は δ-FCost ではない．$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c}(t),t)$ なる $\mathrm{p_c}(t)=t/v$ が存在する場合も同様である． □

以下に示す補題 4 により，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_l}(t),t)$ を満たす，すなわち自身から最右葉を取り除いた木と制約付き δ-出現マッチする木 t について，別の枝刈りを適用することが可能である．この枝刈りは，右＋中央木刈りに対する先読みととらえることができる．

補題 4（左木刈り） $\neg\mathrm{path}(t)$ かつ $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_l}(t),t)$ を満たす木 t に対し（図 5 の (iii)），$t'=t\cdot v$，$\mathrm{d}(v)<\mathrm{d}(\mathrm{rl}(t))$ なる木，すなわち，t の最右葉より浅い位置に節点 v を追加して得られる木 t' を枝刈りすることは，δ-FCost の列挙に影響を与えない．

証明 条件より，t' の葉の数は 3 以上である．また補題 1 より，$\mathrm{p_c}(t')=t'/\mathrm{rl}(t)$ に対し $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c}(t'),t')$ が成り立つ．よって，t' に対して補題 3 が適用される． □

以上の議論に基づき，FreqT と補題 1，3 および 4 による枝刈りを用いた，深さ優先探索による δ-FCost 発見アルゴリズム $\mathrm{dFCost}^{\mathrm{DF}}$（**図 6**）を提案する．

$\mathrm{dFCost}^{\mathrm{DF}}$ は，まず右＋中央木刈りが適用可能か確認し（$\mathrm{dFCost}^{\mathrm{DF}}$-Enum の 1–2 行目），ついで t が単調制約を満たし $\mathrm{dM}_D^{\delta,\mathrm{C}}(t)=\emptyset$ であれば，t を δ-FCost として出力する（3 行目）．このとき，$\mathrm{dFCost}^{\mathrm{DF}}$-Enum を呼び出す前に $\mathrm{C}_A(t)$ は確認されているので，改めて逆単調制約を満たすか確認する必要はない．また左木刈りは，生成する候補木の最右葉の深さを限定することで実現されている（4–6 行目）．

$\mathrm{dFCost}^{\mathrm{DF}}$ に対し，以下の定理が成り立つ．

定理 1 大きさ 1 の頻出順序木の集合 $\mathcal{F}_1$ とデータベース D，許容ノイズ数 δ，最小支持度 σ，単調制約 C_M および逆単調制約 C_A が与えられたとき，$\mathrm{dFCost}^{\mathrm{DF}}$ は，すべての δ-FCost を重複なく列挙する．

証明 FreqT に基づく列挙方法と補題 1，補題 3，補題 4 より明らか． □

Algorithm $\mathrm{dFCost}^{\mathrm{DF}}(\mathcal{F}_1,\ D,\ \sigma,\ \delta,\ \mathrm{C}_M,\ \mathrm{C}_A)$

```
1: for each t ∈ F_1
2:   if C_A(t) then
3:     dFCost^DF-Enum(t, D, σ, δ, C_M, C_A)
4: end for
```

Function $\mathrm{dFCost}^{\mathrm{DF}}$-Enum$(t,\ D,\ \sigma,\ \delta,\ \mathrm{C}_M,\ \mathrm{C}_A)$

```
1: if ¬path(t) ∧ OM_D^{δ,C_M}(p_r(t), t) then return
2: if ∃p_c(t) s.t. OM_D^{δ,C_M}(p_c(t), t) then return
3: if C_M(t) ∧ dM_D^{δ,C}(t)=∅ then output t
4: if ¬path(t) ∧ OM_D^{δ,C_M}(p_l(t), t) then
5:   m := d(rl(t)) else m := = 1
6: for each d (m ≤ d ≤ d(rl(t)) + 1)
7:   for each l ∈ L
8:     s := t · (d, l)
9:     if sup(D, s) ≥ σ ∧ C_A(s) then
10:      dFCost^DF-Enum(s, D, σ, δ, C_M, C_A)
11:  end for
12: end for
```

図 6 δ-FCost 発見アルゴリズム $\mathrm{dFCost}^{\mathrm{DF}}$
Fig. 6 Pseudo code of $\mathrm{dFCost}^{\mathrm{DF}}$.

4.2 幅優先探索による δ-FCost の発見

これまでに，幅優先探索を用いた頻出順序木発見アルゴリズム AMIOT [8] が提案されている．**図 7** に AMIOT の概要を示す．AMIOT は，直列木 t から $C(t)=\{t\cdot(\mathrm{d}(\mathrm{rl}(t))+1,l)\,|\,l\in\mathcal{L}\}$ を生成する**直列木拡張**（AMIOT-Enum の 9 行目）と，$\mathrm{p_r}(t)=\mathrm{p_l}(s)$ なる 2 つの木 t，s から $t\cdot\mathrm{rl}(s)=\mathrm{ll}(t)\bullet s$ を生成する**左右木結合**（5–7 行目）を繰り返し適用することで，すべての頻出順序木を重複なく列挙する[8]．たとえば図 1 では，t_3，t_4 は $\mathrm{p_l}(t_3)=\mathrm{p_l}(t_4)$ なる木に対する直列木拡張により生成され，t_0 は t_8，t_{11} の左右木結合により生成される．なお左右木結合は，その定義から分かるように，木 t の最右枝上に節点を加える操作であると同時に，木 s の最左枝上に節点を加える操作である．したがって，1 つの木は左右両方の方向に拡張される．図 3 (b) に，AMIOT における木 t の拡張の概要を図示する．

Algorithm AMIOT ($\mathcal{F}_1$, D, σ)

1: AMIOT-Enum($\mathcal{F}_1$, D, σ)

Function AMIOT-Enum(F, D, σ)

1: **for each** $t \in F$
2: 　**output** t
3: **end for**
4: $C := \emptyset$
5: **for each** $t, s \in F$ s.t. $\mathrm{p_r}(t) = \mathrm{p_l}(s)$
6: 　$C := C \cup \{t \cdot \mathrm{rl}(s)\}$
7: **end for**
8: **for each** $t \in F$ s.t. path(t)
9: 　$C := C \cup \{t \cdot (\mathrm{d}(\mathrm{rl}(t)) + 1, l) \mid l \in \mathcal{L}\}$
10: **end for**
11: $F' := \{c \in C \mid sup(D, c) \geq \sigma\}$
12: **if** $F' \neq \emptyset$ **then** AMIOT-Enum(F', D, σ)

図 7　頻出順序木発見アルゴリズム AMIOT
Fig. 7　Pseudo code of AMIOT.

ここで，幅優先探索に基づく効率的な δ-FCost 発見アルゴリズムを実現するために，AMIOT に対し，補題 1 に基づく枝刈りを導入することを考える（**図 8** 参照）．

補題 5（中央木刈り）　単調制約 C_M，データベース D，許容ノイズ数 δ と木 t に対し，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c}(t), t)$ を満たす $\mathrm{p_c}(t) = t/v$ が存在するとき（図 8 の (i)），t を枝刈りすることは，δ-FCost の列挙に影響を与えない．

証明　条件より，木 t は直列木ではない．AMIOT は，直列木ではない木 t に対し，最右枝または最左枝上に節点を追加することで拡張を行う．一方条件より，v は t の最右葉・最左葉のいずれでもないので，t から得られる木 t^* は，必ず $t^* \overset{-v}{\succ} t$ を満たす．したがって，t の枝刈りは，$t^* \overset{-v}{\succ} t$ なる木 t^* 以外の木の生成に影響を与えない．また $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c}(t), t)$ と補題 1 より，t^* は δ-FCost ではない．□

もし木 t が，自身から最右または最左葉以外の葉を取り除くことで得られる木と制約付き δ-出現マッチするならば，t から得られる $t^* \overset{-v}{\succ} t$ は δ-FCost ではなく，また $t^* \overset{-v}{\succ} t$ 以外の木の生成に影響を与えないので，t に対して枝刈りを適用することが可能となる．

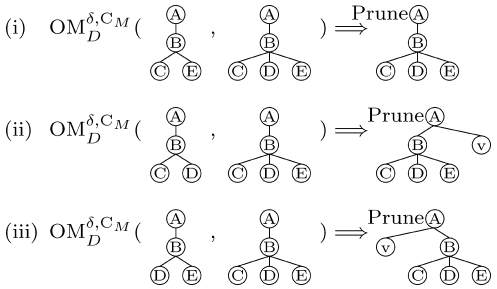

図 8　$\mathrm{dFCost}^{\mathrm{BF}}$ における枝刈りの例
Fig. 8　Pruning in $\mathrm{dFCost}^{\mathrm{BF}}$.

補題 5 を用い，中央木刈りに対する先読みとして，以下の 2 つの枝刈りを導入する．

補題 6（左木刈り）　木 t が，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_l}(t), t)$ かつ $\neg\mathrm{path}(t)$ を満たすとき（図 8 の (ii)），$t' = t \cdot v$，$\mathrm{d}(v) < \mathrm{d}(\mathrm{rl}(t))$ なる木，すなわち，t の最右葉より浅い位置に節点 v を追加して得られる木 t' を枝刈りすることは，δ-FCost の列挙に影響を与えない．

証明　補題 1 より，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(t'/\mathrm{rl}(t), t')$ が成り立つ．したがって，t' に対して補題 5 が適用される．□

補題 7（右木刈り）　木 t が，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_r}(t), t)$ かつ $\neg\mathrm{path}(t)$ を満たすとき（図 8 の (iii)），$t' = v \bullet t$，$\mathrm{d}(v) < \mathrm{d}(\mathrm{ll}(t))$ なる木，すなわち t の最左葉より浅い位置に節点 v を追加して得られる木 t' を枝刈りすることは，δ-FCost の列挙に影響を与えない．

証明　補題 1 より，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(t'/\mathrm{ll}(t), t')$ が成り立つ．したがって，t' に対して補題 5 が適用される．□

以上の議論に基づき，AMIOT と補題 1 および補題 5–7 による枝刈りを用いた，幅優先探索による δ-FCost 発見アルゴリズム $\mathrm{dFCost}^{\mathrm{BF}}$（**図 9**）を提案する．$\mathrm{dFCost}^{\mathrm{BF}}$ は，中央木刈りを用いて F を更新し（$\mathrm{dFCost}^{\mathrm{DF}}$-Enum の 1 行目），その後 F 中の各木 t に対して，$\mathrm{C}_M(t)$ と $\mathrm{dM}_D^{\delta,\mathrm{C}}(t) = \emptyset$ を確認することで δ-FCost のみを出力する（2–4 行目）．また左右木結合の際，左木刈り（7 行目），右木刈り（8 行目）をそれぞれ確認し，両者に枝刈りされない場合にのみ，$t \cdot \mathrm{rl}(s) = \mathrm{ll}(t) \bullet s$ を候補集合 C に追加する（9 行目）．

$\mathrm{dFCost}^{\mathrm{BF}}$ に対し，以下の定理が成り立つ．

定理 2　大きさ 1 の頻出順序木の集合 $\mathcal{F}_1$ とデータベース D，許容ノイズ数 δ，最小支持度 σ，単調制約 C_M および逆単調制約 C_A が与えられたとき，$\mathrm{dFCost}^{\mathrm{BF}}$ は，すべての δ-FCost を重複なく列挙する．

Algorithm $\mathrm{dFCost}^{\mathrm{BF}}(\mathcal{F}_1,\, D,\, \sigma,\, \delta,\, \mathrm{C}_M,\, \mathrm{C}_A)$

1: $\mathcal{F} := \{t \in \mathcal{F}_1 \mid \mathrm{C}_A(t)\}$
2: $\mathrm{dFCost}^{\mathrm{BF}}$-Enum($\mathcal{F}$, D, σ, δ)

Function $\mathrm{dFCost}^{\mathrm{BF}}$-Enum($F$, D, σ, δ, C_M, C_A)

1: $F := F \setminus \{t \in F \mid \exists \mathrm{p_c}(t)\ s.t.\ \mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c}(t), t)\}$
2: **for each** $t \in F$
3: **if** $\mathrm{C}_M(t) \wedge \mathrm{dM}_D^{\delta,\mathrm{C}}(t) = \emptyset$ **then output** t
4: **end for**
5: $C := \emptyset$
6: **for each** $t, s \in F$ s.t. $\mathrm{p_r}(t) = \mathrm{p_l}(s)$
7: **if** $\neg(\neg\mathrm{path}(t) \wedge \mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_l}(t), t) \wedge \mathrm{d}(\mathrm{rl}(s)) < \mathrm{d}(\mathrm{rl}(t)))$
8: $\wedge\ \neg(\neg\mathrm{path}(s) \wedge \mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_r}(s), s) \wedge \mathrm{d}(\mathrm{ll}(t)) < \mathrm{d}(\mathrm{ll}(s)))$
9: **then** $C := C \cup \{t \cdot \mathrm{rl}(s)\}$
10: **end for**
11: **for each** $t \in F$ s.t. $\mathrm{path}(t)$
12: $C := C \cup \{t \cdot (\mathrm{d}(\mathrm{rl}(t)) + 1, l) \mid l \in \mathcal{L}\}$
13: **end for**
14: $F' := \{c \in C \mid sup(D, c) \geq \sigma,\ \mathrm{C}_A(c)\}$
15: **if** $F' \neq \emptyset$ **then**
16: $\mathrm{dFCost}^{\mathrm{BF}}$-Enum($F'$, D, σ, δ, C_M, C_A)

図 9　δ-FCost 発見アルゴリズム $\mathrm{dFCost}^{\mathrm{BF}}$
Fig. 9　Pseudo code of $\mathrm{dFCost}^{\mathrm{BF}}$.

証明　AMIOT に基づく列挙方法と，枝刈りに関する補題 1，補題 5–7 より明らか．　□

4.3　深さ優先＋幅優先探索による δ-FCost の発見

これまでに，深さ優先＋幅優先探索を用いた頻出順序木発見アルゴリズム FreqTDB [11] が提案されている．**図 10** にアルゴリズム FreqTDB を示す．FreqTDB は，木 t と木の集合 $T = \{s \mid \mathrm{p_l}(s) = \mathrm{p_l}(t)\}$ から，新たな候補木の集合

$$C = C_1 \cup C_2$$
$$C_1 = \{t \cdot (\mathrm{d}(\mathrm{rl}(t)) + 1, l) \mid l \in \mathcal{L}\}$$
$$C_2 = \{t \cdot \mathrm{rl}(s) \mid s \in T, \mathrm{d}(\mathrm{rl}(s)) \leq \mathrm{d}(\mathrm{rl}(t))\}$$

Algorithm FreqTDB ($\mathcal{F}_1$, D, σ)

1: FreqTDB -Enum($\mathcal{F}_1$, D, σ)

Function FreqTDB -Enum(F, D, σ)

1: **for each** $t \in F$
2: **output** t
3: $C := \emptyset$
4: **for each** $s \in F$ s.t. $\mathrm{d}(\mathrm{rl}(s)) \leq \mathrm{d}(\mathrm{rl}(t))$
5: $C := C \cup \{t \cdot \mathrm{rl}(s)\}$
6: **end for**
7: $C := C \cup \{t \cdot (\mathrm{d}(\mathrm{rl}(t)) + 1, l) \mid l \in \mathcal{L}\}$
8: $F' := \{c \in C \mid sup(D, c) \geq \sigma\}$
9: **if** $F' \neq \emptyset$ **then** FreqTDB -Enum(F', D, σ)
10: **end for**

図 10　頻出順序木発見アルゴリズム FreqTDB
Fig. 10　Pseudo Code of FreqTDB.

を生成する．ここで，(1) 最右葉の下に節点を加えることで C_1 を得る操作（FreqTDB -Enum の 7 行目）を**最右葉拡張**，(2) 最右葉以外が同一である木どうしを結合することで C_2 を得る操作（4–6 行目）を**兄弟木結合**と呼ぶ．たとえば図 1 において，t_7 は t_6 に対する最右葉拡張により，t_0 は t_{10} と t_{11} の兄弟木結合により生成される．FreqTDB は，最右葉拡張と兄弟木結合を繰り返し適用することで，すべての頻出順序木を重複なく列挙する [11]．

兄弟木結合において，$t \cdot \mathrm{rl}(s) = \mathrm{p_l}(s) \cdot \mathrm{rl}(t) \cdot \mathrm{rl}(s)$ であることに注意が必要である．すなわちこの操作は，t の最右枝に節点を加えると同時に，s に右から 2 番目の葉になる節点を追加する操作ととらえることができる．図 3 (c) に，FreqTDB における木 t の拡張の概要を図示する．先述したとおり，t より得られる任意の木は，t の最右枝上，もしくは右から 2 番目の葉として節点を加えることで得られることに注意が必要である．

深さ優先＋幅優先探索に基づく効率的な δ-FCost 発見アルゴリズムを実現するため，FreqTDB に対し，以下の 3 つの補題を導入する（**図 11** 参照）．

補題 8（右＋中央右木刈り）　単調制約 C_M，データベース D，許容ノイズ数 δ に対し，$\neg\mathrm{path}(t)$ なる木 t が $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_r}(t), t)$，$\mathrm{p_r}(t) = t/v$（図 11 の (i)）または $\exists \mathrm{p_c^r}(t) = t/v$ $s.t.$ $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c^r}(t), t)$（図 11 の (ii)）を満たすとき，$t$ を枝刈りすることは，δ-FCost の列

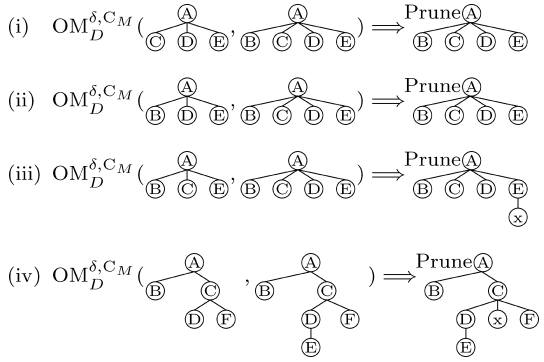

図 11　$\mathrm{dFCost}^{\mathrm{DB}}$ における枝刈りの例
Fig. 11　Pruning in $\mathrm{dFCost}^{\mathrm{DB}}$.

挙に影響を与えない．

証明 $\neg\mathrm{path}(t)$，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_r}(t),t)$，$\mathrm{p_r}(t)=t/v$ の場合を考える．FreqTDB は，最右もしくは右から 2 番目の葉になる位置に新たな節点を追加することで拡張を行う．また条件より，v は t の最右葉でも，$\mathrm{p_l}(t)=t/\mathrm{rl}(t)$ の最右葉（右から 2 番目の葉）でもないので，t から得られる木 t^* は，必ず $t^* \overset{-v}{\succ} t$ を満たす．したがって，t の枝刈りは，$t^* \overset{-v}{\succ} t$ なる木 t^* 以外の木の生成に影響を与えない．一方 $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_r}(t),t)$ と補題 1 より，t^* は δ-FCost ではない．$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c^r}(t),t)$ なる $\mathrm{p_c^r}(t)=t/v$ が存在する場合も同様．□

補題 1 と補題 8 により，木 t が，最右葉または右から 2 番目の葉以外の葉を取り除いた木と制約付き δ-出現マッチするならば，t に対し，枝刈りを適用することが可能であることが分かる．

また木 t が，右から 2 番目の葉を取り除いた木と制約付き δ-出現マッチする場合，以下に示す補題により，別の形式での枝刈りを適用することが可能である．これらは，補題 8 に対する先読みである．

補題 9（中央左木の左刈り）　$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c^l}(t),t)$ を満たす $\mathrm{p_c^l}(t)=t/v$ が存在する木 t に対し（図 11 の (iii)），ある節点を x とし，$t'=t\cdot x$ なる木を枝刈りすることは，δ-FCost の列挙に影響を与えない．

証明 $t'=t\cdot x$ より，$\mathrm{p_c^r}(t')=\mathrm{p_c^l}(t)\cdot x$ が成り立つ．また，補題 1 より $\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c^r}(t'),t')$ であるので，t' に対して補題 8 が適用される．□

補題 10（中央左木の右刈り）　$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c^l}(t),t)$ を満たす $\mathrm{p_c^l}(t)=t/v$ が存在する木 t に対し（図 11 の (iv)），ある節点を x とし，$t'=\mathrm{p_l}(t)\cdot x\cdot \mathrm{rl}(t)$，$\mathrm{d}(x)<\mathrm{d}(\mathrm{rl}(\mathrm{p_l}(t)))$ なる木，すなわち t の 2 番目の葉より浅い位置に，新たな木の 2 番目の葉として x を追加することで得られる木を枝刈りすることは，δ-FCost の列挙に影響を与えない．

証明 補題 1 と条件より，t' に対して，$\mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c^r}(t'),t')$ が成り立つ $\mathrm{p_c^r}(t')=\mathrm{p_l}(\mathrm{p_c^l}(t))\cdot x\cdot\mathrm{rl}(t)$ が存在するので，t' に対して補題 8 が適用される．□

以上の議論に基づき，FreqTDB と補題 1 と補題 8–10 による枝刈りを用いた，深さ優先＋幅優先探索による δ-FCost 発見アルゴリズム $\mathrm{dFCost}^{\mathrm{DB}}$（**図 12**）を提案する．$\mathrm{dFCost}^{\mathrm{DB}}$ において，右＋中央右木刈りは $\mathrm{dFCost}^{\mathrm{DB}}$-Enum の 1 行目，中央左木の左刈りは 3 行目，中央左木の右刈りは 7 行目でそれぞれ実現されている．

Algorithm $\mathrm{dFCost}^{\mathrm{DB}}(\mathcal{F}_1,\ D,\ \sigma)$
1: $\mathcal{F} := \{t \in \mathcal{F}_1 \mid \mathrm{C}_A(t)\}$
2: $\mathrm{dFCost}^{\mathrm{DB}}$-Enum($\mathcal{F}$, D, σ, C_M, C_A)

Function $\mathrm{dFCost}^{\mathrm{DB}}$-Enum($F$, D, σ, C_M, C_A)
1: $F := F \setminus \{t \in F \mid (\neg\mathrm{path}(t) \wedge \mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_r}(t),t)) \vee \exists \mathrm{p_c^r}(t)\ s.t.\ \mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c^r}(t),t)\}$
2: **for each** $t \in F$
3:　**if** $\exists \mathrm{p_c^l}(t)\ s.t.\ \mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c^l}(t),t)$ **then continue**
4:　**if** $\mathrm{C}_M(t) \wedge \mathrm{dM}_D^{\delta,\mathrm{C}}(t)=\emptyset$ **then output** t
5:　$C := \emptyset$
6:　**for each** $s \in F$ s.t. $\mathrm{d}(\mathrm{rl}(s)) \le \mathrm{d}(\mathrm{rl}(t))$
7:　　**if** ($\exists \mathrm{p_c^l}(s)\ s.t.\ \mathrm{OM}_D^{\delta,\mathrm{C}_M}(\mathrm{p_c^l}(s),s) \wedge \mathrm{d}(\mathrm{rl}(t)) < \mathrm{d}(\mathrm{rl}(\mathrm{p_l}(s)))$) **then**
8:　　**continue**
9:　　$C := C \cup \{t\cdot \mathrm{rl}(s)\}$
10:　**end for**
11:　$C := C \cup \{t\cdot(\mathrm{d}(\mathrm{rl}(t))+1, l) \mid l \in \mathcal{L}\}$
12:　$F' := \{c \in C \mid sup(D,c) \ge \sigma, \mathrm{C}_A(c)\ \}$
13:　**if** $F' \neq \emptyset$ **then**
14:　　$\mathrm{dFCost}^{\mathrm{DB}}$-Enum($F'$, D, σ, C_M, C_A)
15: **end for**

図 12　δ-FCost 発見アルゴリズム $\mathrm{dFCost}^{\mathrm{DB}}$
Fig. 12　Pseudo Code of $\mathrm{dFCost}^{\mathrm{DB}}$.

$\mathrm{dFCost}^{\mathrm{DB}}$ に対し，以下の定理が成り立つ．

定理 3　大きさ 1 の頻出順序木の集合 $\mathcal{F}_1$ とデータベース D，許容ノイズ数 δ，最小支持度 σ，単調制約 C_M および逆単調制約 C_A が与えられたとき，$\mathrm{dFCost}^{\mathrm{DB}}$ は，すべての δ-FCost を重複なく列挙する．

証明 FreqTDB に基づく列挙方法と，補題 1，補題 8–10 より明らか．　□

4.4　各提案アルゴリズムの比較

本論文で提案した 3 種のアルゴリズムは，それぞれ異なる頻出パターン発見アルゴリズムに枝刈り手法を導入する形で構成されている．したがってその計算量は，基本的に基となったアルゴリズムのそれを踏襲すると考えられる．FreqT の計算量は文献 1) や 6) に示されている．一方，AMIOT や FreqTDB の列挙戦略は，制限付きの最右拡張ととらえることができ[11]，その観点から，計算量は FreqT に従うものであると考えられる．

ここで，より限定的ではあるが，ある木 t から生成される候補数に関して簡単に考察する．$\mathrm{dFCost}^{\mathrm{DF}}$ では，最大で，最右葉の深さ × ラベル種数（$\mathrm{d}(\mathrm{rl}(t)) \times |\mathcal{L}|$）の候補が生成される可能性がある．したがって，候補数は，ラベル種と最右葉の高さの両方に依存する．一方 $\mathrm{dFCost}^{\mathrm{BF}}$ では，木 t に対し，$t \cdot \mathrm{rl}(s)$ $s.t.$ $\mathrm{p_r}(t) = \mathrm{p_l}(s)$ が生成される．加えて t が直列木の場合は，ラベル種数の直列木が生成される．したがって，生成される候補数は，最右葉の高さには依存せず，同じ大きさの頻出パターンの数に依存すると考えられる．またラベル種数の増加に対する影響も小さいといえる．$\mathrm{dFCost}^{\mathrm{DB}}$ に関しても同様であり，直感的には，$t \cdot \mathrm{rl}(s)$ $s.t.$ $\mathrm{p_l}(t) = \mathrm{p_l}(s)$ と，最右葉（$\mathrm{rl}(t)$）の下にノードを追加した木がラベル種数だけ生成されるので，生成される候補数は，同じ大きさの頻出パターン数に依存する．またラベル種数の増加に対する影響は，$\mathrm{dFCost}^{\mathrm{DF}}$ よりは小さいが $\mathrm{dFCost}^{\mathrm{BF}}$ よりは大きいと考えられる．

次に，枝刈り手法について比較を行う．$\mathrm{dFCost}^{\mathrm{BF}}$ と $\mathrm{dFCost}^{\mathrm{DF}}$ の枝刈りの差異は，最左葉の高さによるもののみである．したがって $\mathrm{dFCost}^{\mathrm{BF}}$ では，非頻出な候補木の生成を抑制したうえでの効果的な枝刈りが期待される．一方 $\mathrm{dFCost}^{\mathrm{DB}}$ は，最右葉を利用した枝刈りができないため，他のアルゴリズムと比較し，一般的には枝刈りの効果が低いと考えられる．

次に，制約付き δ-出現マッチングや $\mathrm{dM}_D^{\delta,\mathrm{C}}(t)$，$\mathrm{rM}_D^{\Delta,\mathrm{C}}(t)$ の確認における差異について考察する．$\mathrm{dFCost}^{\mathrm{BF}}$ では，木 t が生成された時点で，それより一般的な木 $t' \preceq t$ は生成済みであるので，それを利用することができる．これに対し $\mathrm{dFCost}^{\mathrm{DF}}$ や $\mathrm{dFCost}^{\mathrm{DB}}$ では，必要に応じて未だ生成されていない $\mathrm{P}(t)$ の要素を生成，評価する必要がある．ところで，$t' = t \cdot v$ に対する $\mathrm{P}(t')$ の各要素 s' は，$s' = s \cdot v$，$s \in \mathrm{P}(t)$ を満たす．すなわち s' は，t の各親 s に節点 v を追加することで得ることが可能である．したがって，メモリ使用量は増大するが，木 t に対してその親の集合 $\mathrm{P}(t)$ を保持すれば，$t \cdot v$ に対する $\mathrm{P}(t \cdot v)$ の生成，評価は容易である．ところで，木 t の枝刈りや極小性検査に利用される $s \in \mathrm{P}(t)$ の数は，高さやラベル種数に依存せず，最大で，t の葉の数+1 となる．ここで，“+1” は，根ノードを取り除いた木を考慮するためである．

5. 評価実験

提案手法の有効性を評価するため，Java 言語を用いて各提案手法を実装し，PC（CPU: Intel(R) Core2Quad 2.4 GHz，メインメモリ 4 GB）上で評価実験を行った．実装の差異による影響を軽減するために，各実装間で可能な限り共通のモジュールを利用している．また各実装において，最右拡張の最適化手法の 1 つである辺スキップ[1] を導入している．

実験には，Tree Generator[16] を利用して生成した 2 つの合成データ $10K$，$50K$ と，地理データ（mondial）[10]，Web アクセス履歴データ（cslogs）[16] の 2 つの実データを用いた．データセットの概要を**表 1** に示す．詳細は各論文を参照されたい．

実験では，探索空間の大きさを左右する 2 つの要因である (1) 頻度と (2) 制約を考慮し，以下の 2 種類の設定で，抽出された δ-FCost および Δ-TCost の数を計測した．また実データに関しては，実行時間および生成された候補パターン数も計測した．

実験 1　各データセットに対し，制約を与えずに最小支持度を下げ，かつ許容ノイズ数（以下 δ）・許容ノイズ率（以下 Δ）を変化させる．実験結果を**表 2** および**表 4** に示す．

実験 2　実データを対象に，いくつかの単調・逆単調制約の下で，δ および Δ を変化させる．実験結果を**表 3** および**表 5**，**表 6** に示す．

表 4–表 6 において，DF，BF，DB は，それぞれ $\mathrm{dFCost}^{\mathrm{DF}}$，$\mathrm{dFCost}^{\mathrm{BF}}$，$\mathrm{dFCost}^{\mathrm{DB}}$ を表す．また Naive は，各実装から枝刈り機能を省いた場合を表す．なお枝刈りを行わない

表 1　実験で利用したデータセットの概要
Table 1　Overview of datasets used in experiments.

	$\lvert D\rvert$	$\lvert\mathcal{L}\rvert$	S	H	F
10 K	10,000	100	15.0/143	6.0/10	1.5/7
50 K	50,000	100	10.7/143	4.8/10	1.5/7
mondial	955	14,346	122.8/4,859	2.9/5	1.8/197
cslogs	59,691	13,355	12.9/428	3.4/85	2.5/403

$|D|$：データベースの大きさ，$|\mathcal{L}|$：ラベル種数，S：木のサイズの平均/最大，H：木の高さの平均/最大，F：分岐数の平均/最大

表 2　実験 1 の結果：抽出されたパターン数
Table 2　Result of experiments 1: number of extracted patterns.

	δ-FCost			Δ-TCost		
σ	δ			Δ		
10 K	0	3	5	1.1	1.2	1.3
50	9,148	5,868	5,005	4,012	2,960	2,454
25	22,387	12,677	10,417	11,093	7,821	6,244
10	70,665	30,985	22,547	41,111	28,612	21,866
50 K	0	5	10	1.1	1.2	1.3
250	3,046	1,872	1,535	1,003	832	729
100	9,386	5,008	4,061	3,287	2,598	2,224
50	21,254	10,511	8,321	8,396	6,455	5,435
mondial	0	5	10	1.1	1.2	1.3
200	46	25	25	25	25	25
180	109	36	33	32	32	29
160	242	53	47	44	37	34
cslogs	0	5	10	1.1	1.2	1.3
125	2,540	2,412	2,295	2,188	1,953	1,779
100	4,439	3,884	3,534	3,412	2,950	2,681
75	9,925	7,105	6,011	6,136	5,003	4,383

場合，実行速度や候補パターン数などは，δ や Δ，単調制約によらず基本的に一定である．また表中で，太字は各パラメータ設定で最良の結果を，‘–’ はメモリ不足により結果が得られなかった場合を表す．一方，表 3 および表 5，表 6 の C_M-C_A 列は，与えられた制約を表す．S が大きさ，L が葉の数，H が高さを意味し，続く 2 つの数字はその最小値と最大値である．たとえば，“S2-15” は，木の大きさが 2 以上 15 以下でなければならないという制約を表す．また，“L3-10” は木の葉の数が 3 以上 10 以下でなければならないという制約を，“H4-10” は木の高さが 4 以上 10 以下でなければならないという制約を，それぞれ表す．なお，mondial データは高さの最大値が小さいため，高さの制約に関する実験は行っていない．

まず，単純な頻出パターン発見に対する，提案手法の効果について考察する．制約および極小性を考慮しない単純な頻出パターン発見を行った場合，これらのデータセットからそれぞれ，$10K$（$\sigma = 25$）：約 32 万，$50K$（$\sigma = 100$）：約 12 万，mondial（$\sigma = 180$）：約 13 万，cslogs（$\sigma = 100$）：約 1046 万の頻出パターンが発見される．これに対し，抽出された極小パターン（$\sigma = 0$ の δ-FCost）数は，それぞれ $10K$：約 7.7%，$50K$：約 7.8%，mondial：約 0.1%，cslogs：約 0.05%程度であり，抽出されるパターン数の大幅な削減に成功していることが分かる．

表 3　実験 2 の結果：抽出されたパターン数
Table 3　Result of experiments 2: number of extracted patterns.

	δ-FCost			Δ-TCost		
C_M-C_A	δ			Δ		
mondial $\sigma = 180$	0	5	10	1.1	1.2	1.3
S2-15	104	33	30	29	29	29
S3-15	199	147	147	147	147	147
S4-15	686	648	648	648	648	648
L2-10	208	142	142	142	142	139
L3-10	786	606	606	606	606	605
L4-10	2,508	1,826	1,826	1,826	1,826	1,826
cslogs $\sigma = 100$	0	5	10	1.1	1.2	1.3
S3-30	2,364	1,904	1,631	1,579	1,303	1,169
S4-30	1,794	1,422	1,246	1,212	1,018	986
S5-30	1,309	1,133	1,031	1,023	1,014	1,013
H2-10	869	699	632	615	555	514
H3-10	206	171	162	156	147	142
H4-10	108	84	81	80	79	78
L2-10	2,546	1,754	1,445	1,375	1,044	8,82
L3-10	4,009	1,670	1,379	1,303	1,005	8,77
L4-10	11,204	30,56	2,756	2,689	2,492	2,278

次に，抽出された δ-FCost および Δ-TCost の数について考察する．表 2 および表 3 より，δ や Δ が大きくなるほど，抽出された δ-FCost および Δ-TCost の数が減少していることが分かる．この傾向は，特に（制約を考慮しない）低頻度の場合において顕著である．これは，低頻度を用いた頻出パターン発見では，本質的に頻度の差が小さいパターンが数多く抽出される，すなわちノイズの影響が受けやすいということに起因していると考えられる．またこれらの結果は，ノイズを考慮することで，抽出すべきパターンのさらなる限定が達成されていることを表している．

一方，制約を考慮した mondial や葉の数を考慮した cslogs では，探索空間を限定するためにより厳しい単調制約を与えたとしても，逆に抽出されるパターン数が増大してしまっている．これは，単調制約によって排除された探索空間に，多くの δ-FCost や Δ-TCost が存在していたことによるものと考えられる．すなわち，単調制約を厳しくすることで，より緩い制約では δ-FCost や Δ-TCost であったパターンが制約を満たすことができなくなり，結果として，それらと同様の支持度を持つ複数のパターンが，新たな代表元として抽出され

表 4　実験 1 の結果：実データに対する実行時間と生成された候補パターン数
Table 4　Result of experiments 1: execution time and number of candidate patterns.

		実行時間（秒）							候補パターン数（10,000 個）						
	σ	δ			Naive	Δ			δ			Naive	Δ		
mondial		0	5	10		1.1	1.2	1.3	0	5	10		1.1	1.2	1.3
DF		**0.7**	**0.6**	**0.6**	37.5	**0.7**	**0.6**	**0.7**	1.6	**1.5**	**1.5**	42.7	**1.5**	**1.5**	**1.5**
BF	200	1.1	1.0	1.0	16.3	0.9	0.9	0.9	**1.6**	1.5	1.5	4.2	1.5	1.5	1.5
DB		0.9	0.8	0.8	26.0	0.8	0.8	0.9	1.6	1.6	1.6	8.8	1.6	1.6	1.6
DF		**1.1**	**0.8**	**0.7**	–	**0.8**	**0.8**	**0.8**	1.9	**1.6**	**1.5**	–	**1.5**	**1.5**	**1.5**
BF	180	2.8	2.0	1.8	–	1.7	1.5	1.3	**1.7**	1.6	1.6	–	1.6	1.6	1.6
DB		1.7	1.3	1.3	–	1.4	1.4	1.5	1.9	1.8	1.8	–	1.7	1.7	1.7
DF		**2.0**	**1.0**	**0.9**	–	**1.0**	**0.8**	**0.8**	2.5	**1.6**	**1.6**	–	**1.6**	**1.5**	**1.5**
BF	160	6.9	3.3	3.0	–	2.9	2.1	1.7	**2.0**	1.7	1.7	–	1.7	1.6	1.6
DB		4.1	1.8	1.7	–	1.9	1.9	1.8	2.4	1.9	1.8	–	1.8	1.8	1.8
cslogs		0	5	10		1.1	1.2	1.3	0	5	10		1.1	1.2	1.3
DF		31.4	32.2	**31.7**	40.3	34.0	31.8	30.4	8.6	8.4	8.1	62.9	8.1	7.7	7.5
BF	125	33.3	33.5	34.3	33.5	31.1	**31.0**	30.5	**5.1**	**5.1**	**5.1**	7.5	**5.1**	**5.1**	**5.1**
DB		**28.7**	**32.0**	33.7	37.7	**30.0**	32.7	**29.8**	5.9	5.9	5.8	16.7	5.8	5.8	5.8
DF		32.4	33.8	34.4	–	**31.5**	**31.4**	**35.2**	18.8	15.5	13.9	–	13.9	12.7	12.0
BF	100	**32.3**	32.7	34.9	–	33.6	35.5	36.4	**7.1**	**7.0**	**7.0**	–	**7.0**	**7.0**	**6.9**
DB		32.9	**32.1**	**33.8**	–	32.0	34.2	35.5	9.7	9.6	9.5	–	9.5	9.4	9.3
DF		39.0	39.7	49.0	–	42.1	**34.0**	37.3	66.2	40.1	33.0	–	36.4	30.2	25.6
BF	75	38.7	**36.7**	**39.1**	–	**36.1**	39.6	37.7	**13.1**	**12.3**	**12.0**	–	**12.1**	**11.9**	**11.7**
DB		**35.9**	39.1	48.9	–	38.9	40.9	**37.2**	30.0	27.5	26.7	–	27.0	26.1	25.6

たと考えられる．この結果は，代表元のみを抽出することを考えた場合，制約を与えることが必ずしもパターン数の減少にはつながらないことを表しており，統合アプローチに基づく手法のさらなる研究の必要性を示唆するものであると考えられる．

次に，枝刈りの効果について考察する．表 4 と表 5，表 6 から分かるように，枝刈り手法を適用した場合，単純な方法では解が得られない問題に対しても適切な時間で解を得ることに成功している．これらの結果から，各提案手法において，制約付き δ-出現マッチングによる枝刈りが有効に機能していると結論付けることができる．

また，制約を与えない場合（表 4），抽出されたパターン数と同様，δ や Δ を大きくするほど実行時間や生成された候補パターン数が減少する傾向が認められる．しかし制約を考慮した場合（表 5，表 6）は，δ や Δ の増大に対してそれに対応する十分な枝刈りが行われておらず，結果として実行時間の向上が認められない場合も少なくない．このことは，データセットの性質による部分もあるとは考えられるが，提案する枝刈り手法の 1 つの限界を表すものであるといえる．

最後に，提案手法間の比較を行う．生成された候補パターン数に関しては，多くの場合において $\mathrm{dFCost}^{\mathrm{BF}}$ が最も少ない．これは，左右木結合による非頻出候補パターンの抑制によるところが大きいと考えられる．その一方で，候補パターン数の削減に対して実行時間が削減されていないのは，左右木結合のためのコストによるものであると考えられる．$\mathrm{dFCost}^{\mathrm{DF}}$ では，結合相手を検索する必要はない．また $\mathrm{dFCost}^{\mathrm{DB}}$ では，結合相手となる同じ親を持つ木の集合をアルゴリズム中で特別な操作なしに得ることができる．これに対し $\mathrm{dFCost}^{\mathrm{BF}}$ では，同じ大きさを持つ木の集合を対象に結合相手を検索する必要があり，比較的大きな計算コストが必要とされると考えられる．以上の考察と実験結果から，提案手法では，基となった 3 種のアルゴリズムの性質を損なわない形での枝刈り手法の導入が達成されていると考えることができる．

実行時間に関しては，制約なしの mondial では，他の手法と比べ $\mathrm{dFCost}^{\mathrm{DF}}$ が高速に動

表 5　実験 2 の結果：mondial に対する実行時間と生成された候補パターン数
Table 5　Result of experiments 2: execution time and number of candidate patterns in mondial.

		実行時間（秒）							候補パターン数（1,000 個）						
	C_M-C_A	δ			Naive	Δ			δ			Naive	Δ		
mondial, σ=180		0	5	10		1.1	1.2	1.3	0	5	10		1.1	1.2	1.3
DF	S2-15	**1.0**	**0.7**	**0.7**	–	**0.8**	**0.8**	**0.8**	18.6	**15.7**	**15.5**	–	**15.4**	**15.3**	**15.1**
BF		2.9	1.9	1.8	–	1.5	1.5	1.4	**17.1**	16.1	16.0	–	16.0	15.9	15.7
DB		1.8	1.4	1.3	–	1.3	1.4	1.5	19.1	17.6	17.5	–	17.5	17.5	17.4
DF	S3-15	**1.3**	**1.1**	**1.1**	–	**1.3**	**1.2**	**1.2**	20.1	17.6	17.5	–	17.5	17.4	17.4
BF		2.6	1.8	1.7	–	1.6	1.6	1.5	**17.1**	**16.1**	**16.0**	–	**16.0**	**15.9**	**15.7**
DB		1.7	1.2	1.3	–	1.4	1.4	1.4	19.1	17.6	17.5	–	17.5	17.5	17.4
DF	S4-15	**2.6**	**2.5**	2.5	–	2.8	2.9	2.9	30.8	29.2	29.2	–	29.2	29.2	29.1
BF		3.8	3.3	3.4	–	3.2	3.1	2.9	**18.6**	**17.9**	**17.9**	–	**17.9**	**17.8**	**17.7**
DB		2.8	2.5	**2.5**	–	**2.7**	**2.8**	**2.8**	23.6	22.6	22.5	–	22.5	22.4	22.4
DF	L2-10	**1.4**	**1.0**	**1.1**	–	**1.2**	**1.2**	**1.2**	20.7	18.1	18.0	–	18.0	17.9	17.9
BF		2.5	1.7	1.6	–	1.6	1.6	1.4	**17.2**	**16.2**	**16.1**	–	**16.0**	**15.9**	**15.8**
DB		1.7	1.2	1.1	–	1.3	1.2	1.3	19.1	17.6	17.5	–	17.5	17.5	17.4
DF	L3-10	2.8	2.6	2.7	–	3.0	3.0	3.0	36.9	32.9	32.9	–	32.9	32.9	32.9
BF		4.0	3.7	3.1	–	3.2	3.2	3.2	**19.0**	**18.3**	**18.2**	–	**18.2**	**18.2**	**18.1**
DB		**2.6**	**2.3**	**2.4**	–	**2.6**	**2.7**	**2.7**	25.2	24.6	24.6	–	24.5	24.5	24.5
DF	L4-10	7.7	7.9	7.4	–	8.1	8.1	8.4	99.8	86.8	86.8	–	86.8	86.8	86.8
BF		9.8	9.6	9.4	–	9.5	9.6	10.2	**25.0**	**24.6**	**24.6**	–	**24.6**	**24.6**	**24.6**
DB		**6.3**	**6.2**	**6.2**	–	**6.7**	**6.8**	**6.9**	44.9	44.6	44.6	–	44.6	44.6	44.6

作している．また逆に，$\text{dFCost}^{\text{BF}}$ は他手法と比較し多くの計算時間を必要としている．この状況は制約ありの mondial においても類似しており，$\text{dFCost}^{\text{DF}}$ と $\text{dFCost}^{\text{DB}}$ が，同程度の回数最も早く動作しているのに対し，$\text{dFCost}^{\text{BF}}$ は 1 度も最速とはなっていない．一方，cslogs に関してはこのような状況は見られない．制約なしの場合では，手法間に大きな差異は見られないが，制約を考慮した場合には，$\text{dFCost}^{\text{BF}}$ が最も効率的に動作する場合が多く，むしろ mondial の場合とは逆に状況が確認できる．必ずしも実験数が十分ではないが，これらの実験結果は，各手法の得手・不得手に関して，以下の知見を示していると考えられる．まず，mondial のような，頻出パターンに対する極小パターンの割合が大きい，すなわち類似する頻出パターンが少ないという意味で比較的疎なデータに対しては，深さ優先探索に基づく $\text{dFCost}^{\text{DF}}$ や $\text{dFCost}^{\text{DB}}$ が適していると考えられる．また特に，比較的高さの低い木が多い場合は，$\text{dFCost}^{\text{DF}}$ が適していると考えられる．これは，$\text{dFCost}^{\text{DF}}$ が生成する候補パターン数は木の高さ（最右葉の深さ）に依存する，ということにも整合する．一方，cslogs のような比較的密なデータに対しては，$\text{dFCost}^{\text{BF}}$ が適していると考えられる．これは，AMIOT などの幅優先探索に基づくアルゴリズムが元々持つ性質に起因するところも大きいが，$\text{dFCost}^{\text{BF}}$ における枝刈り手法が，それらを阻害しない形で導入されていることも，大きな要因の 1 つであると考えられる．ところで $\text{dFCost}^{\text{DB}}$ に関しては，深さ優先の $\text{dFCost}^{\text{DF}}$ と幅優先の $\text{dFCost}^{\text{BF}}$ の両方の性質を持ち合わせると考えられるが，今回の実験結果からは，$\text{dFCost}^{\text{BF}}$ よりもむしろ $\text{dFCost}^{\text{DF}}$ に近いと考えられる．

6. ま と め

本論文では，制約付きノイズ許容極小順序木パターン発見問題について考察し，制約付き δ-出現マッチングに基づく枝刈りを用いた 3 種のアルゴリズムを提案した．また，合成データおよび実データを用いた実験を通じ，その有効性を示した．

今後の課題としては，(1) 無順序木やグラフなど，より複雑な構造データに対する提案手法の適用や，(2) 提案手法のノイズを考慮した極大元発見への応用などがあげられる．また，(3) ラベルとともに節点に数値を保持するような木構造データに対する提案手法の拡張も，

表 6　実験 2 の結果：cslogs に対する実行時間と生成された候補パターン数
Table 6　Result of experiments 2: execution time and number of candidate patterns in cslogs.

		実行時間（秒）							候補パターン数（1,000 個）						
	C_M-C_A		δ		Naive		Δ			δ		Naive		Δ	
cslogs, σ=100		0	5	10		1.1	1.2	1.3	0	5	10		1.1	1.2	1.3
DF	S3-30	33.6	**32.4**	33.5	–	**31.3**	37.7	**31.9**	189.0	155.4	139.5	–	139.5	127.4	121.3
BF		**31.9**	**32.4**	**33.3**	–	33.9	34.8	34.5	**70.7**	**70.2**	**69.9**	–	**69.9**	**69.5**	**69.4**
DB		33.1	33.0	33.6	–	34.9	**30.8**	32.9	97.2	95.9	94.7	–	94.7	93.7	93.1
DF	S4-30	33.6	35.9	33.5	–	36.7	36.1	36.4	191.6	160.6	148.5	–	148.5	140.0	136.8
BF		**31.8**	36.5	**31.5**	–	**31.1**	34.3	**31.0**	**70.8**	**70.4**	**70.2**	–	**70.2**	**69.9**	**69.8**
DB		33.4	**30.6**	32.7	–	35.0	**33.8**	35.4	97.3	96.2	95.3	–	95.3	94.4	94.0
DF	S5-30	36.6	36.5	33.6	–	**30.4**	**31.7**	35.8	204.4	188.7	184.2	–	184.2	183.2	183.1
BF		34.6	34.7	**30.8**	–	32.8	32.2	32.3	**71.2**	**71.1**	**71.0**	–	**71.0**	**71.0**	**71.0**
DB		**32.0**	**30.9**	32.9	–	32.0	36.9	**30.7**	98.0	97.5	97.0	–	97.0	96.9	96.9
DF	H2-10	33.1	32.4	**33.8**	51.4	36.0	33.4	**31.0**	198.7	176.4	168.5	1,354.7	168.5	163.9	161.8
BF		**30.3**	**32.2**	34.2	35.5	**32.1**	34.5	33.4	**67.5**	**67.4**	**67.3**	122.7	**67.3**	**67.3**	**67.3**
DB		31.8	34.8	35.0	45.7	35.4	**31.1**	33.8	83.2	82.7	82.4	250.9	82.4	82.3	82.2
DF	H3-10	37.9	34.3	34.2	51.4	33.2	33.0	34.8	257.2	255.4	254.4	1,354.7	254.4	253.4	252.8
BF		33.5	36.0	**31.5**	35.5	**30.5**	**31.8**	**31.7**	**67.9**	**67.9**	**67.9**	122.7	**67.9**	**67.9**	**67.9**
DB		**29.7**	**30.7**	31.8	45.7	30.7	35.6	32.6	84.4	84.4	84.4	250.9	84.4	84.3	84.3
DF	H4-10	36.7	36.1	**34.8**	51.4	37.8	**33.5**	**32.8**	262.7	261.3	261.0	1,354.7	261.0	261.0	260.7
BF		31.9	**31.2**	35.0	35.5	**32.2**	33.9	33.4	**68.2**	**68.2**	**68.2**	122.7	**68.2**	**68.2**	**68.2**
DB		**30.3**	35.9	35.5	45.7	35.1	35.8	33.3	85.2	85.2	85.2	250.9	85.2	85.2	85.2
DF	L2-10	47.3	38.6	34.0	–	**32.6**	43.4	32.9	265.0	197.0	181.4	–	181.4	171.1	166.2
BF		31.5	34.5	35.2	–	33.9	34.6	34.2	**72.9**	**72.4**	**72.1**	–	**72.1**	**71.7**	**71.5**
DB		**30.8**	**30.9**	**32.2**	–	32.9	**32.2**	**31.6**	97.2	95.9	94.7	–	94.7	93.7	93.1
DF	L3-10	44.2	38.2	38.9	–	37.7	52.9	60.4	701.3	468.3	460.2	–	460.2	453.9	450.2
BF		37.5	**35.8**	**34.1**	–	**35.3**	**35.7**	**36.8**	**102.8**	**102.6**	**102.4**	–	**102.4**	**102.1**	**102.0**
DB		**36.8**	36.2	38.4	–	36.3	45.1	40.2	207.1	206.2	205.5	–	205.5	204.7	204.3
DF	L4-10	92.7	72.1	76.8	–	82.8	73.9	78.7	2,454.2	1,634.8	1,627.5	–	1,627.5	1,625.9	1,624.6
BF		**49.0**	**49.6**	**47.5**	–	**50.0**	**52.2**	**54.3**	**204.7**	**204.6**	**204.5**	–	**204.5**	**204.5**	**204.5**
DB		74.1	74.8	61.1	–	69.0	65.1	73.2	589.4	589.1	588.7	–	588.7	588.6	588.6

興味深い課題の 1 つである．本論文では，木の形状に関する制約を扱ったが，パターン中の節点が持つ数値の最小（最大）値や，合計の最小（最大）値など，ラベルとは異なる側面からの制約を考慮できるように提案手法を拡張することで，意味的な側面を反映し，より応用に適したパターンマイニングの実現が期待できると考えている．

謝辞　有益なご指摘とご指導を賜りました査読者の方々に深く感謝いたします．本研究の一部は，文部科学省科学研究費補助金（若手研究（B）：課題番号 19700146，特定領域研究：課題番号 19024055，基盤研究（B）：課題番号 20300038）による．

参 考 文 献

1) Asai, T., Abe, K., Kawasoe, S., Arimura, H., Sakamoto, H. and Arikawa, S.: Efficient Substructure Discovery from Large Semi-structured Data, *Proc. 2nd SIAM International Conference on Data Mining* (2002).
2) Boulicaut, J.-F., Bykowski, A. and Rigotti, C.: Free-Sets: A Condensed Representation of Boolean Data for the Approximation of Frequency Queries, *Data Mining and Knowledge Discovery*, Vol.7, No.1, pp.5–22 (2003).

3) Bringmann, B., Zimmermann, A., Raedt, L.D. and Nijssen, S.: Don't Be Afraid of Simpler Patterns, *Proc. 15th European Conference on Machine Learning* (*ECML 2006*), pp.55–66 (2006).
4) Cheng, H., Yuz, P.S. and Han, J.: AC-Close: Efficiently Mining Approximate Closed Itemsets by Core Pattern Recovery, *Proc. 6th IEEE International Conference on Data Mining*, pp.839–844 (2006).
5) Cheng, J., Ke, Y. and Ng, W.: δ-Tolerance Closed Frequent Itemsets, *Proc. 6th IEEE International Conference on Data Mining* (*ICDM '06*), pp.139–148 (2006).
6) Chi, Y., Nijssen, S., Muntz, R.R. and Kok, J.N.: Frequent Subtree Mining — An Overview, *Fundamenta Informaticae*, Vol.66, No.1-2, pp.161–198 (2005).
7) Chi, Y., Xia, Y., Yang, Y. and Muntz, R.R.: Mining Closed and Maximal Frequent Subtrees from Databases of Labeled Rooted Trees, *IEEE Trans. Knowledge and Data Engineering*, Vol.17, No.2, pp.190–202 (2005).
8) 比戸将平，河野浩之：頻出順序木の高速なマイニングアルゴリズム，電子情報通信学会和文論文誌，Vol.J89, pp.163–171 (2006).
9) Li, J., Li, H., Wong, L., Pei, J. and Dong, G.: Minimum Description Length Principle: Generators Are Preferable to Closed Patterns, *Proc. AAAI' 2006*, pp.409–415 (2006).
10) May, W.: Information Extraction and Integration with FLORID: The MONDIAL Case Study, Technical report, Institut für Informatik, Universität Freiburg (1999).
11) 尾崎知伸，大川剛直：制限付き最右拡張を用いた効率的な飽和順序木の発見，情報処理学会論文誌：データベース，Vol.48, No.SIG11 (TOD34), pp.118–127 (2007).
12) Ozaki, T. and Ohkawa, T.: Efficiently Mining Closed Constrained Frequent Ordered Subtrees by Using Border Information, *Proc. 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (*PAKDD'07*), pp.745–752 (2007).
13) Wang, C., Zhu, Y., Wu, T., Wang, W. and Shi, B.: Constraint-Based Graph Mining in Large Database, *Proc. 7th Asia-Pacific Web Conference*, pp.133–144 (2005).
14) Washio, T. and Kok, J.N. and De Raedt, L. (Eds.): *Advances in Mining Graphs, Trees and Sequences*, IOS Press (2005).
15) Yan, X. and Han, J.: CloseGraph: Mining Closed Frequent Graph Patterns, *Proc. 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.286–295 (2003).
16) Zaki, M.J.: Efficiently Mining Frequent Trees in a Forest, *Proc. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.71–80 (2002).
17) Zhu, F., Yan, X., Han, J. and Yu, P.S.: gPrune: A Constraint Pushing Framework for Graph Pattern Mining, *Proc. 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (*PAKDD'07*), pp.388–400 (2007).

(平成 20 年 6 月 9 日受付)
(平成 20 年 10 月 8 日採録)

(担当編集委員　樋口 健)

尾崎　知伸

1973 年生．1996 年慶應義塾大学総合政策学部卒業．1998 年同大学大学院政策・メディア研究科前期修士課程修了．2002 年同研究科講師．2005 年神戸大学大学院自然科学研究科助手．現在，神戸大学自然科学系先端融合研究環助教．博士（政策・メディア）．帰納論理プログラミング，構造データマイニング等の研究に従事．人工知能学会会員．

大川　剛直（正会員）

1963 年生．1986 年大阪大学工学部通信工学科卒業．1988 年同大学大学院工学研究科通信工学専攻博士前期課程修了．大阪大学助手，講師，助教授を経て，2005 年神戸大学大学院自然科学研究科教授．現在，神戸大学大学院工学研究科教授．博士（工学）．知的ソフトウェア，バイオインフォマティクス等の研究に従事．IEEE，人工知能学会，電子情報通信学会，電気学会等の会員．