限定的出現マッチを利用した逆単調制約付き頻出飽和順序木の発見

Mining Frequent Closed Ordered Subtrees under Anti-monotone Constraints by using Restricted Occurrence Matching

尾崎 知伸
Tomonobu Ozaki

神戸大学 自然科学系先端融合研究環
Organization of Advanced Science and Technology, Kobe University
tozaki@cs.kobe-u.ac.jp, <http://www25.cs.kobe-u.ac.jp/~tozaki/>

大川 剛直
Takenao Ohkawa

神戸大学 大学院工学研究科
Graduate School of Engineering, Kobe University
ohkawa@kobe-u.ac.jp, <http://www25.cs.kobe-u.ac.jp/~ohkawa/>

keywords: semi-structured data mining, ordered subtrees, closed patterns, constrained patterns, pruning

Summary

As semi-structured data is used widely in several fields, the importance of structured data mining is increasing recently. Although mining frequent patterns in structured data is one of the most fundamental tasks, frequent pattern miners often discover huge number of patterns. To overcome this problem, two major approaches, condensed representation mining and constraint-based mining, have been proposed. In this paper, as a technique for integrating these two approaches, we propose three algorithms, `RCLOCOT`, `posCLOCOT` and `negCLOCOT`, for discovering closed ordered subtrees under anti-monotone constraints about the structure of patterns to be discovered. The proposed algorithms discover closed constrained subtrees efficiently not by post-processing but by pruning and skipping the search space based on the occurrence matching and the patterns on the border.

Experiments with artificial and real world data were conducted to assess the effectiveness of the proposed algorithms. The results of experiments show that the proposed algorithms succeed in reducing both of the number of candidate subtrees and the execution time compared with a naive algorithm.

1. は じ め に

近年、頻出パタン発見を中心に、構造データを対象としたマイニング手法が注目され、数多くの研究が行われている [Chi 05b, Washio 05]。単純な頻出パタン発見では、大量のパタンが発見されてしまうという問題が指摘されているが、この問題に対して (1) 飽和パタンに代表される頻出パタンの代表元のみを発見する [Chi 05a, Yan 03, Shiozaki 06], (2) 利用者により与えられる制約を満たすパタンのみを発見する [Pei 02, Wang 05] などのアプローチが提案されている。両アプローチは、結果として発見されるパタン数を減少させるという意味では同じであるが、後者は制約を通じて積極的に求めるべきパタンを限定しているのに対し、前者は同じ内容の圧縮表現を求めており、その目的は全く異なる。従って、両アプローチを統合することにより、より効果的なパタン発見が実現されることが期待できる。この考えに基づき、近年、アイテム集合を対象とした両者の統合アプローチ [Bonchi 04, Jia 03] が提案されている。

本論文では、構造データに対する統合アプローチの一つとして、順序木 [Chi 05a] を対象に、逆単調制約付き頻出飽和順序木の発見について議論する。すなわち、節点数や高さ上限などのパタンの形状に関する制約を導入することにより求めるべきパタンを限定し、その上での圧縮表現を求めることを考える。この問題を解決するために、本論文では、出現マッチ [Chi 05a] と制約付き飽和順序木の列挙との関係に対する考察を通じ、3 種の制約付き頻出飽和順序木発見アルゴリズム `RCLOCOT` 及び `posCLOCOT`, `negCLOCOT` を提案する。`RCLOCOT` は、制約なしの飽和順序木発見手法 [Chi 05a] の素直な拡張であり、制約に関する位置限定出現マッチに基づく枝刈りを採用している。一方、`posCLOCOT` 及び `negCLOCOT` は、順序限定出現マッチと境界パタンに基づく枝刈りを用いることで、制約付き飽和順序木の効率的な発見を実現している。またすべてのアルゴリズムは、後処理としてではなく、枝刈りを伴う探索の過程で解の発見を行うという特徴を持つ。

以下に本論文の構成を示す。2 章で用語の準備を行い、

本論文で扱う問題の形式的な定義を与える．3章で，提案するアルゴリズムの出発点として，制約なしの頻出飽和順序木発見アルゴリズムを導入する．4章で RCLOCOT を，次いで5章で posCLOCOT 及び negCLOCOT をそれぞれ提案する．また6章で関連研究について言及する．7章で評価実験の結果を示し，最後に8章でまとめを行う．

2. 準備

2.1 順序木

$\mathcal{L} = \{l_0, l_1, \dots\}$ を，全順序 $\prec$ が与えられているラベルの集合とする． $\mathcal{L}$ 上のラベル付き順序木とは，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 $E \subseteq V^2$ は節点間の親子関係を， $S \subseteq V^2$ は節点間の兄弟関係をそれぞれ表す．また $L: V \rightarrow \mathcal{L}$ はラベル関数である．ここで， $\mathcal{L}$ 上のラベル付き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， $\mathcal{L}$ 上のラベル付き順序木を単に順序木と言う．

2つの順序木 $t = (V, E, S, r, L)$, $s = (V_s, E_s, S_s, r_s, L_s)$ に対し，(1) $(v_1, v_2) \in E$ iff $(\phi(v_1), \phi(v_2)) \in E_s$ ，(2) $(v_1, v_2) \in S$ iff $(\phi(v_1), \phi(v_2)) \in S_s$ ，(3) $L(v) = L_s(\phi(v))$ を満たす1対1関数 $\phi: V \rightarrow V_s$ が存在するとき， t を s の誘導部分木 [Chi 05a, Chi 05b] と定義し， $t \prec s$ (または $s \succ t$) と表記する．上記の条件を満たす関数 ϕ により得られる s 中の頂点集合 $\{\phi(v) \in V_s \mid v \in V\}$ を， s における t の出現と定義する．また， s における t の出現の全集合を $\Phi_{t,s}$ と表記する．

順序木 $t = (V, E, S, r, L)$ において，節点 $v \in V$ のラベルを $l(v)$ と表記する．また v の深さ $d(v)$ を根 r から v への経路上の辺の数と定義する． t の大きさを $|t| = |V|$ ，高さを $\max\{d(v) \mid v \in V\}$ と定義する． t の最右葉を $\text{rml}(t)$ と表記する． t から節点 v を取り除いた木を $t \setminus v$ と表記する．また， t のブランクセットを $B(t) = \{t' \succ t \mid |t'| = |t| + 1\}$ としたとき，2つの順序木 t と $t' \in B(t)$ に対し， t' に含まれ t に含まれない節点を t'/t と表記する．

順序木 t を行きがけ順に辿ることによって得られる節点の系列を $v_1, \dots, v_{|t|}$ としたとき， t の深さラベル列 [Asai 02] を

$$\text{dl}(t) = (d(v_1), l(v_1)) \cdots (d(v_{|t|}), l(v_{|t|})) \in (\mathbb{N} \times \mathcal{L})^{|t|}$$

と定義する．2つの順序木 t_1, t_2 の深さラベル列 $\text{dl}(t_1) = (d_1^1, l_1^1) \cdots (d_{|t_1|}^1, l_{|t_1|}^1)$ ， $\text{dl}(t_2) = (d_1^2, l_1^2) \cdots (d_{|t_2|}^2, l_{|t_2|}^2)$ に対し，以下の(1),(2)のいずれかが成り立つとき $t_1 \prec^{\text{lex}} t_2$ (または $t_2 \succ^{\text{lex}} t_1$)，(2) が成り立つとき $t_1 \prec^{\text{pfx}} t_2$ (または $t_2 \succ^{\text{pfx}} t_1$) と，それぞれ表記する．

$$(1) \exists k \text{ s.t. } \forall i (1 \leq i < k) d_i^1 = d_i^2 \wedge l_i^1 = l_i^2 \wedge (d_k^1 > d_k^2 \vee (d_k^1 = d_k^2 \wedge l_k^1 \prec l_k^2))$$

$$(2) \forall i (1 \leq i \leq |t_1|) d_i^1 = d_i^2 \wedge l_i^1 = l_i^2 \wedge |t_1| < |t_2|$$

すなわち， $\prec^{\text{lex}}$ は $\mathcal{T}_{\mathcal{L}}$ 上の全順序関係を，また $t_1 \prec^{\text{pfx}} t_2$ は深さラベル列で t_1 が t_2 の接頭辞であることを表す．

図1 順序木間の関係

図1に順序木の例を示す．図中において， $\text{dl}(t) = (0, A) (1, B) (1, C)$ である．また， $d(\text{rml}(s)) = 2$ ， $l(\text{rml}(s)) = G$ ， $r = s \setminus \text{rml}(s)$ である．一方， $s = s' \setminus (t'/t)$ であり， $t \prec^{\text{lex}} t'$ 及び $t \prec^{\text{pfx}} q$ が成り立つ．

2.2 制約付き頻出飽和順序木

順序木の集合 $D = \{t_1, t_2, \dots, t_N\}$ を順序木データベースと呼ぶ．このとき， D における順序木 t の支持度を $\text{sup}_D(t) = \sum_{s \in D} \text{Occ}_t(s) / N$ と定義する．ここで $\text{Occ}_t(s)$ は， $t \prec s$ なら1，そうでないなら0を返す関数である．データベース D と最小支持度 $\sigma (0 \leq \sigma \leq 1)$ に対し，支持度 σ 以上を前提とし，同じ支持度を持つ順序木集合における極大元，すなわち $\text{sup}_D(t) \geq \sigma \wedge \forall s \succ t (s \neq t \rightarrow \text{sup}_D(t) \neq \text{sup}_D(s))$ を満たす順序木 t を，頻出飽和順序木と定義する．

順序木 t が制約 C を満たすとき， $C(t)$ と書く． t が制約を満たすとき，その任意の誘導部分木 t' も制約を満たす場合，すなわち $\forall t' \prec t, C(t) \rightarrow C(t')$ が成り立つとき， C は逆単調制約であると定義する．例えば， t と $t' \prec t$ に対し $|t| \geq |t'|$ が成り立つので，木の大きさの上限は逆単調制約である．同様に，木の高さや葉の数，最大分岐数や含まれるラベル種の上限なども，逆単調制約である．

順序木データベース D ，最小支持度 σ ，逆単調制約 C に対し， $\text{sup}_D(t) \geq \sigma \wedge C(t) \wedge \forall s \succ t (C(s) \wedge s \neq t \rightarrow \text{sup}_D(t) \neq \text{sup}_D(s))$ が成り立つとき， t は制約付き頻出飽和順序木であると定義する．すなわち制約付き頻出飽和順序木とは，支持度 σ 以上を前提とし，制約を満たす同支持度を持つ順序木集合における極大元である．

以上の準備の下，本論文で扱う問題は，順序木データベース D ，逆単調制約 C 及び最小支持度 σ を入力とし， D 中のすべての制約付き頻出飽和順序木を発見する問題と定式化される．

3. 頻出飽和順序木の発見

本章では，議論の出発点として (制約なしの) 頻出飽和順序木の発見アルゴリズム [Chi 05a] を導入する．

順序木 t から，新たな順序木の集合 $\text{ref}(t) = \{t' \mid \text{dl}(t') = \text{dl}(t) (d, l), d(\text{rml}(t)) + 1 \geq d \geq 1, l \in \mathcal{L}\}$ を得る操作を

図 2 $OM_D(t, t')$ の例

最右拡張 [Asai 02] と呼ぶ。 $dl(t_0) = (0, l), l \in \mathcal{L}$ なる全ての順序木 t_0 に対し、最右拡張を繰り返し適用することで、 $\mathcal{T}_L$ のすべての要素を重複することなく列挙することが可能である [Asai 02]。ここで $s \stackrel{\text{pfx}}{>} t$ なる順序木 s は、 t への最右拡張の繰り返し適用により列挙される。

データベース D と $sup_D(t) \geq \sigma$ なる順序木 t に対し、 t と同じ支持度を持つブランチ中の順序木の集合を、 t のトランザクションマッチ $TM_D(t) = \{t' \in B(t) \mid sup_D(t') = sup_D(t)\}$ と定義する。定義より、 $TM_D(t) = \emptyset$ であれば、 t は頻出飽和順序木である。一方、 D において t の全出現を覆う (含む) 形で $t' \in B(t)$ の出現が存在する [Chi 05a] とし、すなわち $\forall s \in D, \forall V_s \in \Phi_{t,s}, \exists V'_s \in \Phi_{t',s}, V_s \subset V'_s$ が成り立つとき、 $OM_D(t, t')$ と表記する。図 2 に例を示す。図中において、 $D = \{t_a, t_b\}$ における t_x の各出現 $\phi_{t_x} \in \Phi_{t_x, t_D} (D = a, b)$ に対し、 $\phi_{t_x} \subset \phi_{t_y}$ なる t_y の出現 ϕ_{t_y} が存在するので、 $OM_D(t_x, t_y)$ が成り立つ。一方 Φ_{t_z, t_a} には、 $\{\textcircled{B}_5, \textcircled{C}_7\} \in \Phi_{t_z, t_a}$ を含む出現が存在しないため、 $OM_D(t_x, t_z)$ とはならない。

t の出現マッチを $OM_D(t) = \{t' \notin \text{ref}(t) \mid OM_D(t, t')\}$ と定義する。このとき、任意の t 及び $t' \in OM_D(t)$ に対し、

$$\forall q [q \stackrel{\text{pfx}}{>} t \rightarrow \exists q' \stackrel{\text{pfx}}{>} t', q' \in OM_D(q)] \quad (1)$$

が成り立つ [Chi 05a]。例えば図 1 において、 $t' \in OM_D(t)$ と仮定すると、 $r, s \stackrel{\text{pfx}}{>} t$ に対し $r' \in OM_D(r)$ 及び $s' \in OM_D(s)$ が成り立つ。上記の関係は、 t に対して $t' \in OM_D(t)$ が存在する場合、 t の拡張により得られる任意の順序木 $q \stackrel{\text{pfx}}{>} t$ に対し、同支持度を持つより大きな順序木 $q' \in OM_D(q)$ が t' の拡張により得られる ($q' \stackrel{\text{pfx}}{>} t'$) ことを表しており、結果として、 q が飽和順序木になり得ないことを意味している。また $t' \in OM_D(t)$ より $t' \stackrel{\text{pfx}}{<} t$ は成立せず、 t' の列挙に t は影響を与えない。従って、 t の拡張をやめる、すなわち枝刈りを行うことが可能となる。

以上の準備の下、図 3 に出現マッチに基づく枝刈りを伴う最右拡張による頻出飽和順序木発見アルゴリズム C_{LOOT} を示す。なおこのアルゴリズムは、[Chi 05a] で示されたアルゴリズムを、飽和順序木の発見に特化したものである。 C_{LOOT} は、最右拡張を用いて $\stackrel{\text{lex}}{<}$ の順にすべての順序木を列挙する。その際、 $sup_D(t) < \sigma$ また

Algorithm $C_{\text{LOOT}}(D, \sigma)$

```

1: for  $l \in \mathcal{L}$  in order of  $\stackrel{\text{lex}}{<}$ 
2:    $t := (0, l)$ 
3:   if  $sup_D(t) \geq \sigma$  then Check( $t, D, \sigma$ )

```

Procedure Check(t, D, σ)

```

1: if  $OM_D(t) \neq \emptyset$  then return
2: if  $TM_D(t) = \emptyset$  then output( $t$ )
3: Refine( $t, D, \sigma$ )

```

Procedure Refine(t, D, σ)

```

1: for  $d \in \{d(\text{rml}(t)) + 1, \dots, 1\}$ 
2:   for  $l \in \mathcal{L}$  in order of  $\stackrel{\text{lex}}{<}$ 
3:      $t' := t(d, l)$ 
4:     if  $sup_D(t') \geq \sigma$  then Check( $t', D, \sigma$ )

```

図 3 飽和順序木発見アルゴリズム C_{LOOT}

は $OM_D(t) \neq \emptyset$ なる順序木 t は枝刈りの対象となる。また $TM_D(t)$ を計算することで、非飽和な順序木の出力を回避している。

4. 位置限定出現マッチを用いた制約付き頻出飽和順序木の発見

本章では、出現マッチと制約付き飽和順序木との関係について考察し、 C_{LOOT} の素直な拡張である、制約付き頻出飽和順序木発見アルゴリズム RC_{LOCOT} を提案する。

4.1 制約付き出現マッチによる枝刈りの不完全性

データベース D と逆単調制約 C 及び $sup_D(t) \geq \sigma \wedge C(t)$ なる順序木 t に対し、 t と同支持度を持つ制約を満たすブランチ中の順序木の集合を、 t の制約付きトランザクションマッチ $TM_D^C(t) = \{t' \in TM_D(t) \mid C(t')\}$ と定義する。定義より、 $TM_D^C(t) = \emptyset$ であれば t は制約付き頻出飽和順序木である。図 4 に例を示す。今 $D = \{t_d\}$ 、制約を“葉の数は 3 以内” ($C = l \leq 3$) とする。このとき、 s_d に対し節点 $\textcircled{0}$ を追加した順序木 $s_d \cdot \textcircled{0}$ は $TM_D^{l \leq 3}(s_d)$ に含まれるが、順序木 $s_d \cdot \textcircled{1}$ や $s_d \cdot \textcircled{6}$ は、制約を満たさないので $TM_D^{l \leq 3}(s_d)$ に含まれない。

一方、順序木 t の制約付き出現マッチ、すなわち出現マッチに含まれる制約を満たす順序木の集合を、 $OM_D^C(t) = \{t' \in OM_D(t) \mid C(t')\}$ と定義する。例えば図 4 において、制約を“木の高さは 2 以内” ($C = h \leq 2$) としたとき、 $s_d \cdot \textcircled{0}, s_d \cdot \textcircled{5}$ は $OM_D(s_d)$ には含まれるが $OM_D^{h \leq 2}(s_d)$ には含まれない。

逆単調制約を対象とした場合、順序木 s が制約を満たしたとしても、 $s' \in OM_D(s)$ が制約を満たすかは分らない。従って制約を考慮しない場合とは異なり、任意の t と $t' \in OM_D^C(t)$ に対し、制約を導入した式 (1) の拡張

図 4 種々の出現マッチの例

である

$$\forall s [(s \succ^{\text{pfx}} t \wedge C(s)) \rightarrow \exists s' \succ^{\text{pfx}} t', s' \in OM_D^C(s)] \quad (2)$$

が成立するとは限らない．すなわち， t の拡張により得られる制約を満たす任意の順序木 s に対し， t' の拡張により，同支持度を持つ制約を満たすより大きな順序木 $s' \in OM_D^C(s)$ が得られるとは限らず，結果として， $OM_D^C(t) \neq \emptyset$ を基準とした枝刈りを導入することはできない．例えば図 1 において， $t' \in OM_D^C(t)$ が成り立ち，制約を“節点数は 7 以内”($C = v \leq 7$) としたとき， $s \succ^{\text{pfx}} t$ に対し $C(s)$ ではあるが $s' \notin OM_D^{v \leq 7}(s)$ であり，式 (2) は成り立たない．また仮に， $\sup_D(s) \geq \sigma$ であれば s は制約付き飽和順序木となる．従って，たとえ $OM_D^C(t) \neq \emptyset$ であっても， t を枝刈りすることは制約付き飽和順序木 $s \succ^{\text{pfx}} t$ の生成を妨げるので適切ではない．

4.2 位置限定出現マッチを利用した枝刈りの導入

制約付き飽和順序木発見において，出現マッチによる枝刈りを導入するために，制約付き出現マッチに制限を加えることを考える．

制約付き頻出飽和順序木に関し，以下の補題が成り立つ．
 [補題 1] 2つの順序木 $t, t' \in OM_D^C(t)$ に対し，式 (2) が成り立てば， $s \succ^{\text{pfx}} t \wedge C(s)$ なる順序木 s は，制約付き頻出飽和順序木とはならない．

《証明》 式 (2) より，制約を満たす順序木 s に対し， $s' \in OM_D^C(s)$ ，すなわち $s' \succ s$ なる制約を満たす順序木 $s' \succ^{\text{pfx}} t'$ が存在する．従って，定義より， s は制約付き飽和順序木ではない．□

この補題に着目し，順序木 t の位置限定出現マッチを

$$ROM_D^C(t) = \{t' \in OM_D^C(t) \mid t, t' \text{ で式 (2) が成立}\}$$

と定義する．補題 1 より， $t' \in ROM_D^C(t)$ が存在すれば，制約を満たす順序木 $s \succ^{\text{pfx}} t$ は，制約付き飽和順序木とはなり得ない．また， t は t' の列挙に影響を与えない．従って， $ROM_D^C(t) \neq \emptyset$ に基づく t の枝刈りは，探索の完全性に影響を与えない．

なお $ROM_D^C(t)$ は，制約 C と $OM_D^C(t)$ から計算することが可能である．例えば，制約として葉数の上限が与

Algorithm RCLocOT(D, σ, C)

```

1: for  $l \in \mathcal{L}$  in order of  $\prec$ 
2:    $t := (0, l)$ 
3:   if  $C(t) \wedge \sup_D(t) \geq \sigma$  then Check( $t, D, \sigma, C$ )
   Procedure Check( $t, D, \sigma, C$ )
4:   if  $ROM_D^C(t) \neq \emptyset$  then return
5:   if  $TM_D^C(t) = \emptyset$  then output( $t$ )
6:   Refine( $t, D, \sigma, C$ )
   Procedure Refine( $t, D, \sigma, C$ )
7:   for  $d \in \{d(\text{rml}(t)) + 1, \dots, l\}$ 
8:     for  $l \in \mathcal{L}$  in order of  $\prec$ 
9:        $t' := t(d, l)$ 
10:      if  $C(t') \wedge \sup_D(t') \geq \sigma$  then Check( $t', D, \sigma, C$ )

```

図 5 位置限定出現マッチによる枝刈りを伴う制約付き飽和順序木発見アルゴリズム RCLocOT

えられた場合， $ROM_D^C(t) = \{t' \in OM_D^C(t) \mid t' \text{ と } t \text{ の葉の数は同じ}\} = \{t' \in OM_D^C(t) \mid t' \text{ は } t \text{ の根または葉に節点を加えることで得られる順序木}\}$ と計算できる（図 4 参照）．また，制約として高さの上限が与えられた場合は， $ROM_D^C(t) = \{t' \in OM_D^C(t) \mid t'/t \text{ は } t' \text{ の根ではない}\}$ となる．すなわち位置限定出現マッチは，節点 t'/t が制約に影響を与えない t' によって構成される．

以上の議論に基づき，位置限定出現マッチによる枝刈りを導入した制約付き頻出飽和順序木発見アルゴリズム RCLocOT を図 5 に示す．なお RCLocOT は，制約を考慮している以外は CLocOT と全く同じ構造をしており，CLocOT の素直な拡張となっている．

ところで，木の節点数の上限など，与える逆単調制約 C によっては，任意の順序木で位置限定出現マッチが空になる，すなわち $\forall t \in T_{\mathcal{L}} ROM_D^C(t) = \emptyset$ が成り立ち，全く枝刈りが行われないこととなる．この場合，RCLocOT は，最右拡張に基づく木の列挙と $TM_D^C(t)$ に基づく飽和性の検査を繰り返す，最も単純なアルゴリズムと等価となる．

5. 境界パターンを用いた制約付き頻出飽和順序木の発見

4.1 節で述べたように，制約付き頻出飽和順序木発見における枝刈り導入の大きな障害は， $t' \in OM_D^C(t)$ に対し， $s' \succ^{\text{pfx}} t' \wedge s' \in OM_D(s) \wedge s' \notin OM_D^C(s)$ なる順序木 $s \succ^{\text{pfx}} t$ が存在することである．このような s は， s' が制約を満たさないために制約付き頻出飽和順序木となる可能性があり，その (t を通じた) 枝刈りは探索の不完全性の原因となる．本章では，探索中に境界パターンと呼ばれるパターンを保持することで，出現マッチに基づく枝刈りを伴いながらも，上記のような順序木 s を生成する 2 つのアルゴリズム posCLocOT 及び negCLocOT を提案する．

5.1 正境界パタンを利用した枝刈りの導入

データベース D , 逆単調制約 C , 最小支持度 σ に対し, その拡張により制約を満たさない順序木が生成されるような制約を満たす頻出順序木, すなわち

$$\sup_D(r') \geq \sigma \wedge C(r') \wedge \exists s' \in \text{ref}(r'), \neg C(s')$$

を満たす順序木 r' を, 制約を満たす・満たさないの境界にあるという意味で正境界パタンと呼ぶ. また正境界パタンの集合を $\text{Bd}_{D,\sigma}^{C,+}$ と表記する. このとき, 出現マッチに s' を持つ順序木 s は, $\neg C(s')$ により, 不完全性の原因となる $s' \in \text{OM}_D(s) \wedge s' \notin \text{OM}_D^C(s)$ を満たすことに注意が必要である.

図 1 において, $t' \in \text{OM}_D^C(t)$, $r' \in \text{Bd}_{D,\sigma}^{C,+}$, $s' \notin \text{OM}_D^C(s)$ であると仮定する. このとき, t とその制約付き出現マッチ t' 及び正境界パタン r' を利用することで, 以下の方法により, 制約付き出現マッチに基づく枝刈りによる不完全性の原因となる $s \succ^{\text{pfx}} t$ を生成することが可能である.

$$\begin{aligned} t \Rightarrow t' \in \text{OM}_D^C(t) &\Rightarrow \underline{r'} \succ^{\text{pfx}} t', r' \in \text{Bd}_{D,\sigma}^{C,+} \\ &\Rightarrow \underline{r} = r' \setminus (t'/t) \Rightarrow \underline{s} \in \text{ref}(r) \end{aligned}$$

すなわち, t' の深さラベル列を接頭辞として r' を得, r' から節点 t'/t を削除することで r を獲得し, さらに r に最右拡張を適用することで s を生成する. このことは, 正境界パタンを利用することで, 飽和順序木になり得ない $t <^{\text{pfx}} p <^{\text{pfx}} r$ なる順序木 p の生成を回避すると共に, 完全性を保つために必要な s の生成が実現できる可能性を示している. この考えに基づき, 制約付き出現マッチによる t の枝刈りを導入するためには, 以下の 2 点に対する考察が必要となる.

- (1) どの様にして $\text{Bd}_{D,\sigma}^{C,+}$ を準備するのか.
- (2) $s' \in \text{OM}_D(s) \wedge s' \notin \text{OM}_D^C(s)$ なるすべての順序木 $s \succ^{\text{pfx}} t$ を生成する必要がある.

以下, この 2 点について考察する.

正境界パタンは順序木であるので, 前処理として求めるのではなく, 制約付き飽和順序木発見の過程において適宜発見・保持していくことが望ましい. C_{LOOT} に従うと, 順序木は $<^{\text{lex}}$ の順で生成される. また順序木の生成順に関し, 以下の補題が成り立つ.

[補題 2] 2 つの順序木 $t, t' \in \text{OM}_D^C(t)$ が $t' <^{\text{lex}} t$ を満たせば, $\forall r' [r' \succ^{\text{pfx}} t' \rightarrow r' <^{\text{lex}} t]$ が成り立つ. すなわち, t' の拡張により得られる任意の順序木 r' は, t より先に生成される.

《証明》 $t' \in \text{OM}_D^C(t)$ より, $t <^{\text{pfx}} t'$ は成立しない. 従って, $\text{dl}(t')$, $\text{dl}(t)$ に対し, $<^{\text{lex}}$ の条件 (1) が成り立つ. また $r' \succ^{\text{pfx}} t'$, すなわち $\text{dl}(t')$ は $\text{dl}(r')$ の接頭辞であるので, $\text{dl}(r')$, $\text{dl}(t)$ に対し $<^{\text{lex}}$ の条件 (1) が成り立つ. $\square$

この順序木の生成順序に着目し, 順序木 t が生成された時点での制約付き出現マッチ, すなわち t の順序限定出現マッチを

$$\text{POM}_D^C(t) = \{t' \in \text{OM}_D^C(t) \mid t' <^{\text{lex}} t\}$$

と定義する. また, t が生成された時点での正境界パタンの集合を $\text{Bd}_t^+ = \{r' \in \text{Bd}_{D,\sigma}^{C,+} \mid r' <^{\text{lex}} t\}$ と定義する. このとき, 以下の補題 3 が成り立つ. すなわち, $t' \in \text{POM}_D^C(t)$ を前提としたとき, 枝刈り導入の障害となる $s' \succ^{\text{pfx}} t' \wedge s' \in \text{OM}_D(s) \wedge s' \notin \text{OM}_D^C(s)$ を満たす任意の頻出順序木 $s \succ^{\text{pfx}} t$ に対し, s の生成に必要な正境界パタン r' は t 以前に生成され, 結果として s の列挙が保証される.

[補題 3] 順序木 t 及び $t' \in \text{POM}_D^C(t)$ に対し, 以下の式が成り立つ.

$$\begin{aligned} \forall s \succ^{\text{pfx}} t \quad &[(\sup_D(s) \geq \sigma \wedge \\ &\exists s' \succ^{\text{pfx}} t', s' \in \text{OM}_D(s) \wedge s' \notin \text{OM}_D^C(s)) \rightarrow \\ &\exists r' \succ^{\text{pfx}} t', r' \in \text{Bd}_t^+ \wedge r = r' \setminus (t'/t) \wedge r <^{\text{pfx}} s] \end{aligned}$$

《証明》 $s' \in \text{OM}_D(s) \wedge s' \notin \text{OM}_D^C(s)$ なる $s \succ^{\text{pfx}} t$ に対し, $r' \succ^{\text{pfx}} t' \wedge r = r' \setminus (t'/t) \wedge r <^{\text{pfx}} s$ なる $r' \in \text{Bd}_t^+$ が存在することは明らかである. 従って, $r' \in \text{Bd}_t^+$ を示す必要がある. $t' \in \text{POM}_D^C(t)$ より $t' <^{\text{lex}} t$ である. また補題 2 と $r' \succ^{\text{pfx}} t'$ より, $r' <^{\text{lex}} t$ が成り立つ. 従って, 定義より $r' \in \text{Bd}_t^+$ が成り立つ. $\square$

以上の議論に基づき, 順序限定出現マッチと正境界パタンに基づく枝刈りを導入した制約付き頻出飽和順序木発見アルゴリズム posCLOCOT を図 6(左) に示す. ここで, $\text{select}(X)$ は集合 X から任意の要素を一つ取り出す関数, $\min <^{\text{pfx}}(X) = \{x \in X \mid \nexists x' \in X, x' <^{\text{pfx}} x\}$ は $<^{\text{pfx}}$ の意味で極小集合を計算する関数である. RCLOCOT 同様, posCLOCOT でも最右拡張に基づき $<^{\text{lex}}$ の順で木を列挙していく. その際, $\text{POM}_D^C(t) \neq \emptyset$ なる t に対しては, t 自身の展開をやめる (枝刈り) と同時に, select と $\min <^{\text{pfx}}$ を利用して, t の枝刈りに起因する不完全性の回避に必要とされる順序木集合 T を獲得する. すなわち posCLOCOT では, まず枝刈りを適用し, それによって引き起こされる不完全性を回避するために, T 中の各木を $<^{\text{lex}}$ 順に展開するという操作が行われている.

5.2 負境界パタンを利用した枝刈りの導入

本節では, posCLOCOT と同様の考えに基づき, 負境界パタンを利用した制約付き頻出飽和順序木発見アルゴリズム negCLOCOT を提案する.

データベース D , 逆単調制約 C , 最小支持度 σ に対し, 正境界パタンに最右拡張を適用して得られる制約を満たさない頻出順序木, すなわち

$$\sup_D(s') \geq \sigma \wedge \neg C(s') \wedge \forall t' <^{\text{pfx}} s', C(t')$$

を満たす順序木 s' を負境界パタンと呼び, その集合を $\text{Bd}_{D,\sigma}^{C,-}$ と表記する. s' を負境界パタンとしたとき, 不完全性の原因となる $s' \in \text{OM}_D(s) \wedge s' \notin \text{OM}_D^C(s)$ を満たす s が存在する可能性がある. 前節同様, 図 1 において $t' \in \text{OM}_D^C(t)$, $C(s)$, $\neg C(s')$ であると仮定する. このと

Algorithm posCLOCOT(D, σ, C)	Algorithm negCLOCOT(D, σ, C)
1: $Bd := \emptyset$	1: $Bd := \emptyset$
2: for $l \in \mathcal{L}$ in order of $\prec$	2: for $l \in \mathcal{L}$ in order of $\prec$
3: $t := (0, l)$	3: $t := (0, l)$
4: if $C(t) \wedge \sup_D(t) \geq \sigma$ then	4: if $\sup_D(t) \geq \sigma$ then
5: Check(t, D, σ, C, Bd)	5: Check(t, D, σ, C, Bd)
Procedure Check(t, D, σ, C, Bd)	Procedure Check(t, D, σ, C, Bd)
1: if $t \in \text{Bd}_{D,\sigma}^{C,+}$ then $Bd := Bd \cup \{t\}$	1: if $t \in \text{Bd}_{D,\sigma}^{C,-}$ then $Bd := Bd \cup \{t\}$
2: if $POM_D^C(t) \neq \emptyset$ then	2: if $\neg C(t)$ then return
3: $t' := \text{select}(POM_D^C(t))$	3: if $POM_D^C(t) \neq \emptyset$ then
4: $T := \min^{\text{pfx}} < ($	4: $t' := \text{select}(POM_D^C(t))$
5: $\{r' \setminus (t'/t) \in \mathcal{T}_L \mid t' \prec^{\text{pfx}} r', r' \in Bd\}$	5: $T := \min^{\text{pfx}} < ($
6: for $t'' \in T$ in order of $\prec^{\text{lex}}$	6: $\{s' \setminus (t'/t) \in \mathcal{T}_L \mid t' \prec^{\text{pfx}} s', s' \in Bd\}$
7: if $t'' \in \text{Bd}_{D,\sigma}^{C,+}$ then $Bd := Bd \cup \{t''\}$	7: for $t'' \in T$ in order of $\prec^{\text{lex}}$
8: Refine(t'', D, σ, C, Bd)	8: Check(t'', D, σ, C, Bd)
9: return	9: return
10: if $TM_D^C(t) = \emptyset$ then output(t)	10: if $TM_D^C(t) = \emptyset$ then output(t)
11: Refine(t, D, σ, C, Bd)	11: Refine(t, D, σ, C, Bd)
Procedure Refine(t, D, σ, C, Bd)	Procedure Refine(t, D, σ, C, Bd)
1: for $d \in \{d(\text{rml}(t)) + 1, \dots, 1\}$	1: for $d \in \{d(\text{rml}(t)) + 1, \dots, 1\}$
2: for $l \in \mathcal{L}$ in order of $\prec$	2: for $l \in \mathcal{L}$ in order of $\prec$
3: $t' := t(d, l)$	3: $t' := t(d, l)$
4: if $C(t') \wedge \sup_D(t') \geq \sigma$ then	4: if $\sup_D(t') \geq \sigma$ then
5: Check(t', D, σ, C, Bd)	5: Check(t', D, σ, C, Bd)

図 6 順序限定出現マッチと境界パターンを用いた制約付き飽和順序木発見アルゴリズム posCLOCOT(左) 及び negCLOCOT(右)

き、制約付き出現マッチと負境界パターンである s' を利用することで、

$$\begin{aligned} \underline{t} \Rightarrow \underline{t}' \in OM_D^C(t) \Rightarrow \\ \underline{s}' \succ^{\text{pfx}} t', s' \in \text{Bd}_{D,\sigma}^{C,-} \Rightarrow \underline{s} = s' \setminus (t'/t) \end{aligned}$$

の手順により、 t から s を生成することが可能である。

t が生成された時点での負境界パタンの集合を $\text{Bd}_t^- = \{s' \in \text{Bd}_{D,\sigma}^{C,-} \mid s' \prec^{\text{lex}} t\}$ と定義する。このとき、前節と同様の議論から、以下の補題 4 が成り立つ。これにより、 t と $t' \in POM_D^C(t)$ 及び $s' \prec^{\text{lex}} t$ なる負境界パターン s' を利用することで、枝刈り導入の障害となる $s' \in OM_D(s) \wedge s' \notin OM_D^C(s)$ を満たす任意の順序木 $s \succ^{\text{pfx}} t$ の列挙が可能となる。

[補題 4] 順序木 t 及び $t' \in POM_D^C(t)$ に対し、以下の式が成り立つ。

$$\begin{aligned} \forall q \succ^{\text{pfx}} t \quad [(\sup_D(q) \geq \sigma \wedge \\ \exists q' \succ^{\text{pfx}} t', q' \in OM_D(q) \wedge q' \notin OM_D^C(q)) \rightarrow \\ \exists s' \succ^{\text{pfx}} t', s' \in \text{Bd}_t^- \wedge s = s' \setminus (t'/t) \wedge \\ (s = q \vee s \prec^{\text{pfx}} q)] \end{aligned}$$

《証明》 $q' \in OM_D(q) \wedge q' \notin OM_D^C(q)$ なる $q \succ^{\text{pfx}} t$ に対し、 $s' \succ^{\text{pfx}} t' \wedge s = s' \setminus (t'/t) \wedge (s = q \vee s \prec^{\text{pfx}} q)$ なる $s' \in$

$\text{Bd}_{D,\sigma}^{C,-}$ が存在することは明らかである。従って、 $s' \in \text{Bd}_t^-$ を示す必要がある。 $t' \in POM_D^C(t)$ より $t' \prec^{\text{lex}} t$ である。また補題 2 と $s' \succ^{\text{pfx}} t'$ より、 $s' \prec^{\text{lex}} t$ が成り立つ。従って、定義より $s' \in \text{Bd}_t^-$ が成り立つ。□

以上の議論に基づき、順序限定出現マッチと負境界パターンに基づく枝刈りを導入した制約付き頻出飽和順序木発見アルゴリズム negCLOCOT を図 6(右) に示す。negCLOCOT では、負境界パターンを生成・保持するために、制約を満たすか否かのチェックを posCLOCOT とは異なるタイミングで行っている。

順序木 $t' \in POM_D^C(t)$ が与えられたとき、posCLOCOT では、飽和順序木になり得ない順序木 $r = r' \setminus (t'/t)$, $r' \in \text{Bd}_{D,\sigma}^{C,+}$ が生成される。これに対し、negCLOCOT ではこのような r の生成は回避される。その一方で、negCLOCOT では、負境界パターンを得るために制約を満たさない頻出順序木の生成と評価が必要とされる。従って、posCLOCOT と negCLOCOT のどちらが優れているかは自明ではなく、その性能は与えられる問題に依存すると考えられる。

5.3 動的順序の導入

順序限定出現マッチを利用した枝刈りが適用可能か、すなわち順序木 t に対し $POM_D^C(t) \neq \emptyset$ となるか否かは、

$\mathcal{L}$ により構成される $\prec^{\text{lex}}$ に強く依存する．そこで，高頻度な順序木から生成した方が，順序限定出現マッチによる枝刈りが促進されるとのヒューリスティクスに基づき，一種の最適化手法として順序木の動的順序を導入する．

順序木 x, y に対して，全順序 $\prec^d$ を以下の様に定義する．

- (1) $x \prec^{\text{pfx}} y$ であれば $x \prec^d y$
- (2) $x \setminus \text{rml}(x) = y \setminus \text{rml}(y)$ かつ，以下のいずれかの条件が成り立てば $x \prec^d y$

$$\text{a } \text{sup}_D(x) > \text{sup}_D(y)$$

$$\text{b } \text{sup}_D(x) = \text{sup}_D(y) \wedge x \prec^{\text{lex}} y$$

(最右葉を取り除いた順序木同士が同一の場合，まず支持度で比較を行い，その後 $\prec^{\text{lex}}$ による比較を行う．)

- (3) $x' \prec^{\text{pfx}} x, y' \prec^{\text{pfx}} y$ に対し， $x \prec^d y$ であれば $x' \prec^d y'$ ($x \prec^d y$ であれば， x の拡張により列挙される任意の x' は， y の拡張により列挙される任意の y' より早い)

順序木 t の順序限定出現マッチを $POM_D^C(t) = \{t' \in OM_D^C(t) \mid t' \prec^d t\}$ と再定義し，かつ順序木の生成順序を $\prec^d$ に従うものに変更することにより，これまでの議論に影響を与えることなく，提案手法に動的順序を導入することが可能である．

6. 関連研究

これまでに，飽和パタン発見及び制約の導入に関して，数多くの研究が行われている．

飽和パタン発見は，アイテム集合を対象にその研究が進められている [Pei 00a, Uno 03, Uno 04, Wang 03, Zaki 02a] が，近年では，系列データ [Wang 04, Arimura 05] や木構造データ [Chi 05a, Shiozaki 06, 尾崎 07]，グラフ構造データ [Yan 03] など，構造データを対象とした手法が提案されている．制約の導入に関しても同様であり，アイテム集合 [Pei 00b, Bonchi 05] や系列データ [Pei 02] を対象とした手法に加え，構造データを対象とした手法 [Wang 05, Zou 06, Zhu 07] が提案されている．

また，飽和パタン発見と制約の導入の統合アプローチとして，アイテム集合を対象とした制約付き頻出飽和アイテム集合発見アルゴリズムが提案されている [Bonchi 04, Jia 03]．その一方で，構造データを対象とした統合アプローチは，筆者の知る限り未だ提案されていない．また [Bonchi 04] における手法が，後処理として一部の制約付き頻出飽和アイテム集合を発見するのに対し，本論文での提案手法は，いずれも枝刈りを伴う探索の過程ですべての制約付き頻出飽和順序木を発見しており，その点で既存手法と大きく異なる．一方 [Bonchi 04] では，逆単調制約だけでなく，単調制約，すなわち $\forall t' \succ t, C(t) \rightarrow C(t')$ なる性質を持つ制約をも対象としている．本論文では深くは言及しないが，提案手法において枝刈りの対象を単調制約を満たす順序木に限定することで，他に影響を与えることなく単調制約の扱いを可能とするアルゴリズム

表 1 データセットの概要

データ名	$ D $	S	H	L	F	$ \mathcal{L} $
D_{100}	50000	10.7	4.8	4.1	1.5	100
D_{10}	50000	11.1	5.5	4.0	1.4	10
CSLOGS	59691	12.9	3.4	8.2	2.5	13355
Mondial	955	122.8	2.9	53.3	1.8	14346

$|D|$: データベース中のトランザクション数

S: 木のサイズの平均, H: 木の高さの平均

L: 木の葉数の平均, F: 平均分岐数, $|\mathcal{L}|$: ラベル種数

が構築できると考えている．

7. 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 RCLOCOT , posCLOCOT 及び negCLOCOT のそれぞれを実装し，評価実験を行った．なお実装による差異を軽減するため，順序木の頻度計算や $TM_D^C(t)$, $OM_D^C(t)$ の計算など，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．評価には，Tree Generator[Zaki 02b] を利用して生成した 2 つの合成データ (D_{100} , D_{10}) と，Web アクセス履歴データ (CSLOGS)[Zaki 06] 及び地理データ (Mondial)[May 99] の 2 つの実データを用いた．各データの概要を表 1 に示す．詳細は各文献を参照されたい．

実験では，いくつかの制約を対象に，最小支持度 σ を徐々に下げながら実行時間及び生成された候補木の数を計測した．またすべての実験は，Linux マシン (Intel(R) Xeon(TM) 2.8GHz, メインメモリ 2GB) 上で行った．

合成データに対する実験結果を表 2 に示す．表中において，‘pos’ は posCLOCOT ，‘neg’ は negCLOCOT ，‘R’ は RCLOCOT をそれぞれ表す．また ‘N’ は， RCLOCOT から Check の一行目を削除したアルゴリズム，すなわち全く枝刈りを行わなかった場合を表す．表中の ‘pos’ 及び ‘neg’ における括弧内の数字は，動的順序付けを行わなかった場合を表す．さらに，下線は各制約及び頻度毎に最良の結果を，波線は枝刈りを行わない場合と比較し結果の改善が見られなかった場合をそれぞれ表している．

実験結果より，単純なアルゴリズムと比較し，ほとんどの場合において，実行速度の改善が見られる．また頻度が低く，制約が緩い場合に，改善の度合いが高い傾向が確認できる．さらに， D_{100} の葉数上限以外では，動的順序の導入が効果的に働いていることが確認できる．一方， negCLOCOT において，一部生成された候補順序木の数が増加している．これは，負境界パタンを求めるために，制約を満たさないパタンをも列挙しているためである．ところで， posCLOCOT や negCLOCOT では，葉数上限の制約を緩めた場合に，生成される候補順序木数が減少し，実行速度が改善される場合がある．これは，制約を緩めたことにより枝刈りを妨げる要因となっていた $s' \in OM_D(s) \wedge s' \notin OM_D^C(s)$ を満たす順序木 s が減少し，これらに対応する正負の境界パタン数も減少したた

表 2 合成データに対する実験結果
Time: 実行時間 (秒), Cand.: 生成された候補順序木数 (1000 個), [] 内は動的順序を採用した場合

D_{100}		$\sigma = 500/50000$				$\sigma = 250/50000$				$\sigma = 125/50000$				$\sigma = 50/50000$			
		Time		Cand.		Time		Cand.		Time		Cand.		Time		Cand.	
節点数上限																	
10	pos	15.5	[19.0]	18.5	[22.5]	19.2	[23.3]	37.0	[43.9]	22.8	[26.7]	68.9	[79.4]	27.4	[31.3]	154.1	[173.4]
	neg	17.1	[21.0]	22.1	[27.1]	22.1	[27.2]	46.8	[56.4]	27.3	[32.3]	94.1	[110.1]	33.4	[38.7]	230.0	[260.0]
	N		22.8		30.0		28.2		61.3		32.4		113.8		37.6		246.6
15	pos	22.3	[28.2]	30.5	[37.3]	31.9	[40.8]	78.3	[96.7]	44.4	[54.0]	198.4	[236.8]	62.7	[74.9]	654.1	[748.7]
	neg	22.6	[28.8]	30.6	[37.1]	32.3	[40.7]	79.1	[98.8]	44.5	[54.6]	207.8	[249.7]	66.0	[78.3]	729.6	[837.4]
	N		40.3		66.9		58.1		184.0		79.3		476.4		110.7		1520.4
高さ上限																	
5	pos	15.2	[16.6]	19.0	[20.0]	19.0	[20.6]	41.3	[42.9]	23.4	[24.7]	88.8	[91.2]	29.9	[31.6]	242.7	[247.1]
	neg	14.6	[15.9]	17.8	[19.3]	18.3	[20.0]	37.7	[39.5]	22.1	[23.9]	78.1	[79.8]	27.9	[29.6]	205.6	[204.5]
	R, N	18.5	18.8	20.0	25.0	22.2	23.8	39.3	57.5	25.5	29.3	71.9	130.3	30.6	38.1	165.0	381.0
7	pos	24.9	[27.6]	36.4	[38.8]	35.9	[38.8]	96.4	[100.5]	48.5	[51.9]	244.1	[247.1]	71.3	[73.5]	825.9	[807.0]
	neg	22.6	[25.2]	29.9	[31.8]	31.6	[34.6]	74.3	[78.3]	41.4	[44.3]	178.8	[180.1]	59.0	[60.3]	574.1	[535.4]
	R, N	28.7	33.5	33.9	54.5	38.6	48.2	80.4	154.0	48.3	67.0	172.5	421.2	62.2	99.8	468.7	1551.0
葉数上限																	
5	pos	19.3	[18.7]	27.2	[23.8]	28.0	[27.2]	67.6	[62.1]	39.2	[37.4]	182.6	[170.1]	58.1	[55.6]	658.6	[607.0]
	neg	20.5	[18.4]	26.9	[21.9]	28.5	[26.3]	66.5	[55.1]	39.2	[35.4]	173.0	[146.6]	58.7	[52.3]	653.8	[544.6]
	R, N	21.7	41.8	25.9	70.6	31.7	62.9	68.3	200.6	43.1	88.8	168.7	555.4	60.7	131.6	513.1	1913.1
10	pos	20.7	[18.4]	26.7	[21.5]	28.6	[26.0]	64.4	[52.4]	38.5	[33.9]	157.9	[127.4]	55.0	[46.7]	512.8	[397.5]
	neg	20.7	[18.4]	26.7	[21.5]	28.6	[26.0]	64.4	[52.4]	38.5	[33.9]	157.9	[127.4]	54.9	[46.7]	512.8	[397.5]
	R, N	22.1	42.9	27.2	75.2	33.5	66.1	75.4	228.5	46.2	97.8	203.5	688.7	68.9	157.8	720.3	2918.5
D_{10}		$\sigma = 500/50000$				$\sigma = 250/50000$				$\sigma = 125/50000$				$\sigma = 50/50000$			
		Time		Cand.		Time		Cand.		Time		Cand.		Time		Cand.	
節点数上限																	
10	pos	22.7	[23.0]	16.8	[16.9]	25.9	[26.4]	28.1	[28.8]	28.9	[29.6]	48.5	[50.6]	31.8	[32.2]	91.4	[96.3]
	neg	23.1	[23.4]	18.7	[18.8]	27.5	[27.7]	33.6	[34.4]	31.0	[32.1]	61.7	[64.9]	35.3	[36.2]	129.0	[137.6]
	N		29.7		23.8		34.2		40.0		36.4		69.0		39.7		128.0
15	pos	24.5	[25.2]	18.7	[19.0]	29.6	[30.7]	36.2	[37.7]	35.0	[36.5]	75.2	[82.4]	42.4	[45.0]	208.1	[235.6]
	neg	24.4	[25.4]	18.7	[18.9]	29.3	[30.6]	35.6	[37.1]	34.7	[36.3]	73.0	[80.2]	41.5	[44.7]	201.0	[230.1]
	N		34.6		33.5		42.8		75.1		52.3		169.7		65.2		494.5
高さ上限																	
5	pos	23.3	[24.1]	17.4	[17.3]	27.0	[27.9]	30.9	[31.1]	30.8	[31.8]	57.4	[59.2]	35.1	[35.9]	128.6	[136.3]
	neg	21.2	[22.5]	15.9	[15.6]	25.3	[26.1]	27.6	[27.8]	28.7	[29.5]	50.2	[52.3]	32.0	[33.2]	107.0	[117.0]
	R, N	25.4	26.2	17.5	21.1	29.9	30.9	29.3	39.9	32.2	35.5	50.9	78.0	36.2	40.6	101.3	188.9
7	pos	24.6	[25.3]	19.6	[19.3]	30.0	[30.4]	36.5	[37.1]	35.3	[36.4]	72.8	[77.2]	41.3	[43.5]	183.1	[202.0]
	neg	24.2	[25.0]	18.5	[18.4]	28.7	[29.6]	34.0	[35.0]	33.5	[34.9]	65.6	[71.5]	39.4	[41.3]	157.9	[182.3]
	R, N	32.3	33.4	24.3	30.9	38.1	40.5	44.3	66.4	44.4	49.6	83.7	146.7	51.0	60.4	190.1	429.0
葉数上限																	
5	pos	19.5	[19.9]	14.0	[13.5]	23.4	[24.0]	27.6	[27.2]	27.7	[28.2]	57.4	[58.1]	32.9	[33.4]	157.2	[159.9]
	neg	19.5	[19.9]	13.7	[13.3]	23.2	[23.8]	25.9	[25.7]	27.1	[27.9]	51.6	[53.2]	32.0	[33.0]	135.5	[144.0]
	R, N	20.8	34.6	13.6	33.4	24.9	43.7	26.6	74.3	28.4	53.0	54.2	166.4	34.1	66.4	137.2	479.8
10	pos	19.5	[20.1]	13.7	[13.2]	23.5	[24.1]	25.6	[25.4]	26.7	[27.7]	49.6	[51.8]	31.9	[32.7]	121.1	[133.5]
	neg	19.4	[20.1]	13.7	[13.2]	23.5	[24.0]	25.6	[25.4]	26.7	[27.7]	49.6	[51.8]	31.9	[32.8]	121.1	[133.5]
	R, N	20.4	34.6	13.7	33.6	24.5	44.3	27.2	76.1	28.4	53.7	57.0	177.7	34.4	69.5	156.9	574.2

めと考えられる。

実データに対する実験結果を表 3 に示す。人工データの場合と同様、概ね、低頻度かつ緩い制約の場合に、高い改善が見られる。特に、Mondial においては、実行速度が 50 倍以上改善する場合も確認された。これらの結果より、各提案アルゴリズムの有効性が確認できる。

一方 RCLOCOT は、高さ上限においては有効に機能しているが、葉数上限ではあまり枝刈りが行われず、逆にチェックのためのオーバーヘッドにより実行速度が低下していることが分かる。また動的順序の導入に関しては、CSLOGS では有効に働いているが、Mondial では逆効果となってしまう。これらの結果は、問題に適したアルゴリズムや全順序の選択が重要であることを示唆

するものであると考えられる。

8. ま と め

本論文では、順序木データを対象とした制約の導入と飽和パターン発見の統合手法として、出現マッチと境界パターンを利用し、枝刈りを伴う探索の過程ですべての逆単調制約付き頻出飽和順序木を発見する 3 種のアルゴリズムを提案した。また、合成データ及び実データを用いた評価実験を通じ、その有効性を確認した。

今後の課題としては、種々の制約に特化した最適化手法の導入や、無順序木やグラフ構造データなど、より複雑な構造データへの提案手法の適用などがあげられる。

表 3 実データに対する実験結果

Time: 実行時間 (秒), Cand.: 生成された候補順序木数 (1000 個), [] 内は動的順序を採用した場合

CSLOGS		$\sigma = 120/59691$				$\sigma = 110/59691$				$\sigma = 100/59691$			
		Time		Cand.		Time		Cand.		Time		Cand.	
節点数上限													
20	pos	58.5	[71.9]	219.4	[233.6]	76.3	[91.8]	405.8	[410.7]	120.7	[109.2]	630.0	[631.1]
	neg	57.5	[76.2]	215.6	[239.7]	81.5	[107.3]	475.3	[482.0]	138.0	[135.8]	828.3	[830.1]
	N		74.6		276.3		108.6		470.4		151.0		709.8
30	pos	52.8	[87.2]	156.8	[243.5]	58.7	[341.7]	245.2	[789.0]	646.6	[2055.5]	6156.1	[7186.3]
	neg	52.6	[87.2]	156.8	[243.5]	55.7	[345.7]	208.8	[787.8]	659.6	[2572.0]	6106.0	[7927.2]
	N		85.2		320.9		340.9		1379.8		2682.1		11210.1
高さ上限													
7	pos	53.4	[54.6]	163.2	[165.6]	55.3	[56.0]	192.5	[193.7]	82.2	[55.5]	236.1	[233.8]
	neg	53.0	[54.6]	164.8	[168.5]	54.6	[56.0]	195.7	[198.1]	80.8	[55.3]	240.1	[239.0]
	R, N	53.3	54.2	156.9	169.6	54.1	55.2	183.6	203.9	53.5	66.0	222.5	280.2
15	pos	52.6	[84.7]	156.8	[243.5]	131.4	[212.5]	647.5	[767.4]	289.8	[392.2]	1423.1	[1517.5]
	neg	52.5	[87.2]	156.8	[243.5]	71.0	[206.2]	639.4	[849.7]	131.9	[355.0]	1849.3	[2050.0]
	R, N	53.2	85.3	157.1	320.9	54.3	250.1	183.9	892.7	53.7	480.4	222.9	1634.5
葉上限													
7	pos	59.6	[67.7]	203.0	[213.2]	93.8	[109.1]	413.0	[382.5]	268.6	[263.8]	1378.3	[1010.9]
	neg	59.7	[73.5]	193.2	[216.7]	98.6	[140.4]	396.7	[402.4]	346.9	[441.6]	1475.5	[1180.9]
	R, N	116.2	89.8	211.6	266.9	120.9	181.5	375.2	649.0	333.7	516.6	1009.5	2466.3
10	pos	56.1	[83.2]	180.2	[240.6]	102.8	[247.5]	485.4	[675.2]	712.3	[1475.1]	3950.9	[4269.1]
	neg	54.5	[85.5]	167.7	[241.8]	88.5	[289.1]	376.0	[701.3]	692.2	[2122.0]	3279.3	[4877.9]
	R, N	130.5	102.0	243.1	316.8	259.9	333.1	682.6	1217.9	1732.4	2241.0	4309.2	9752.1
節点数 × 高さ × 葉数上限													
20×	pos	63.1	[69.3]	225.2	[227.3]	83.0	[86.7]	394.9	[391.5]	101.0	[104.1]	593.9	[592.4]
15×	neg	62.2	[73.8]	214.0	[227.1]	86.6	[100.0]	449.2	[448.2]	113.5	[127.2]	760.7	[760.7]
7	N		89.9		262.1		134.6		439.7		128.9		657.0
30×	pos	56.8	[81.7]	187.3	[243.7]	130.6	[183.3]	660.4	[725.6]	243.0	[323.7]	1350.1	[1407.0]
15×	neg	55.0	[85.6]	170.0	[242.6]	86.4	[187.8]	660.9	[799.3]	139.1	[316.3]	1752.0	[1887.9]
10	N		101.4		316.8		257.8		842.0		395.9		1518.1

MONDIAL		$\sigma = 200/955$				$\sigma = 175/955$				$\sigma = 160/955$			
		Time		Cand.		Time		Cand.		Time		Cand.	
節点数上限													
15	pos	3.2	[1.3]	98.2	[45.6]	21.9	[14.9]	392.4	[190.4]	46.0	[34.3]	738.6	[443.0]
	neg	3.2	[1.2]	97.8	[44.9]	16.3	[8.4]	337.6	[128.8]	34.1	[21.1]	580.6	[266.6]
	N		12.8		487.8		144.3		5075.6		310.3		9829.9
20	pos	3.3	[1.2]	97.8	[44.8]	12.6	[2.7]	302.1	[86.3]	21.9	[4.2]	453.9	[120.6]
	neg	3.3	[1.2]	97.8	[44.8]	12.6	[2.7]	302.1	[86.3]	21.6	[4.2]	453.8	[120.3]
	N		12.9		487.9		151.2		5103.9		334.1		9946.0
高さ上限													
3	pos	5.0	[3.5]	117.9	[63.4]	45.9	[39.1]	589.5	[354.9]	85.8	[75.3]	1008.1	[643.2]
	neg	3.2	[1.2]	97.8	[44.8]	12.0	[2.6]	302.1	[86.3]	21.0	[4.2]	453.8	[120.3]
	R, N	1.0	12.8	40.4	487.9	1.5	150.4	56.3	5103.9	2.1	331.0	67.0	9946.0
5	pos	3.3	[1.2]	97.8	[44.8]	12.5	[2.7]	302.1	[86.3]	21.7	[4.2]	453.8	[120.3]
	neg	3.3	[1.2]	97.8	[44.8]	12.5	[2.7]	302.1	[86.3]	21.8	[4.2]	453.8	[120.3]
	R, N	1.0	12.9	40.4	487.9	1.5	151.0	56.3	5103.9	2.1	334.2	67.0	9946.0
葉上限													
10	pos	3.9	[3.4]	94.4	[73.8]	53.3	[50.0]	998.9	[907.5]	112.8	[107.1]	2378.7	[2244.3]
	neg	3.0	[2.4]	76.2	[54.0]	41.6	[37.3]	515.6	[399.8]	90.8	[81.9]	1186.3	[1004.8]
	R, N	17.8	12.9	468.5	482.9	184.9	135.6	4682.6	4875.2	352.0	286.3	8586.5	9198.1
15	pos	2.2	[1.2]	68.2	[44.8]	10.8	[3.0]	239.6	[88.3]	22.7	[5.6]	406.0	[133.4]
	neg	2.2	[1.2]	68.2	[44.8]	10.7	[2.7]	238.2	[86.4]	21.7	[4.6]	396.9	[122.3]
	R, N	18.4	13.0	473.2	487.9	215.5	150.4	4886.1	5103.9	440.9	335.1	9233.4	9945.3
節点数 × 高さ × 葉数上限													
15×	pos	2.3	[1.3]	68.9	[45.6]	20.7	[14.8]	332.0	[190.4]	46.5	[34.3]	687.8	[443.0]
5×	neg	2.2	[1.3]	68.3	[44.9]	15.1	[8.5]	275.4	[128.8]	34.6	[20.9]	526.0	[266.6]
15	N		12.8		487.8		144.3		5075.6		318.2		9829.9
20×	pos	4.2	[3.5]	87.6	[63.4]	44.9	[40.6]	514.9	[356.4]	85.2	[74.0]	930.9	[653.4]
3×	neg	2.1	[1.2]	68.2	[44.8]	10.3	[2.6]	238.2	[86.4]	20.8	[4.5]	396.9	[122.3]
15	N		12.7		487.9		149.2		5103.9		333.9		9945.3

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）による。

◇ 参 考 文 献 ◇

- [Arimura 05] Arimura, H. and Uno, T.: A Polynomial Space and Polynomial Delay Algorithm for Enumeration of Maximal Motifs in a Sequence, *Proc. of the 16th Annual International Symposium on Algorithms and Computation (ISAAC'05)*, LNCS 3827, pp.724-737, Springer (2005)
- [Asai 02] Asai, T., Abe, K., Kawasoe, S., Arimura, H., Sakamoto, H. and Arikawa, S.: Efficient Substructure Discovery from Large Semi-structured Data, *Proc. of the 2nd SIAM International Conference on Data Mining (SDM'02)*, pp.158-174 (2002)
- [Bonchi 04] Bonchi, F. and Lucchese, C.: On Closed Constrained Frequent Pattern Mining, *Proc. of the 4th IEEE International Conference on Data Mining (ICDM'04)*, pp.35-42 (2004)
- [Bonchi 05] Bonchi, F., Giannotti, F., Mazzanti, A. and Pedreschi, D.: Exante: A Preprocessing Method for Frequent-Pattern Mining, *IEEE Intelligent Systems*, Vol.20, No.3, pp.25-31 (2005)
- [Chi 05a] Chi, Y., Xia, Y., Yang, Y. and Muntz, R. R.: Mining Closed and Maximal Frequent Subtrees from Databases of Labeled Rooted Trees, *IEEE Transactions on Knowledge and Data Engineering*, Vol.17, No.2, pp.190-202 (2005)
- [Chi 05b] Chi, Y., Nijssen, S., Muntz, R. R. and Kok, J. N.: Frequent Subtree Mining—An Overview, *Fundamenta Informaticae*, Special Issue on Graph and Tree Mining, Vol.66, No.1-2, pp.161-198 (2005)
- [Jia 03] Jia, L., Pei, R. and Pei, D.: Tough constraint-based frequent closed itemsets mining, *Proc. of the 2003 ACM symposium on Applied computing*, pp.416-420 (2003)
- [May 99] May, W.: Information Extraction and Integration with FLORID: The MONDIAL Case Study, Technical Report 131, Institut für Informatik, Universität Freiburg (1999)
- [尾崎 07] 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.SIG 11(TOD34), pp.118-127 (2007)
- [Pei 00a] Pei, J., Han, J. and Mao, R.: Closet: An efficient algorithm for mining frequent closed itemsets, *Proc. of the ACM-SIGMOD International Workshop on Data Mining and Knowledge Discovery (DMKD 2000)*, pp.21-30 (2000)
- [Pei 00b] Pei, J. and Han, J.: Can We Push More Constraints into Frequent Pattern Mining?, *Proc. of 2000 International Conference on Knowledge Discovery and Data Mining*, pp.350-354 (2000)
- [Pei 02] Pei, J., Han, J. and Wang, W.: Mining Sequential Patterns with Constraints in Large Databases, *Proc. of the 11th International Conference on Information and Knowledge Management*, pp.18-25 (2002)
- [Shiozaki 06] Shiozaki, H., Ozaki, T. and Ohkawa, T.: Mining Closed and Maximal Frequent Induced Free Subtrees, *Workshops Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006)*, pp.14-18 (2006)
- [Uno 03] Uno, T., Asai, T., Arimura, H. and Uchida, Y.: LCM: An Efficient Algorithm for Enumerating Frequent Closed Item Sets, *Proc. of International Conference on Data Mining, Frequent Itemset Mining Implementations (FIMI'03)* (2003)
- [Uno 04] Uno, T., Asai, T., Uchida, Y. and Arimura, H.: An efficient algorithm for enumerating closed patterns in transaction databases, *Proc. of the 7th International Conference on Discovery Science*, pp.16-31 (2004)
- [Wang 03] Wang, J., Han, J. and Pei, J.: Closet+: Searching for the best strategies for mining frequent closed itemsets, *Proc. of the 9th ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003)*, pp.236-245 (2003)
- [Wang 04] Wang, J. and Han, J.: BIDE: Efficient Mining of Frequent Closed Sequences, *Proc. of International Conference on Data Engineering (ICDE'04)*, pp.79-90 (2004)
- [Wang 05] Wang, C., Zhu, Y., Wu, T., Wang, W. and Shi, B.: Constraint-Based Graph Mining in Large Database, *Proc. of the Seventh Asia Pacific Web Conference*, pp.133-144 (2005)
- [Washio 05] Washio, T., Kok, J. N. and De Raedt, L. (eds.): *Advances in Mining Graphs, Trees and Sequences*, Vol. 124, *Frontiers in Artificial Intelligence and Applications*, IOS Press (2005)
- [Yan 03] Yan, X. and Han, J.: CloseGraph: Mining Closed Frequent Graph Patterns, *Proc. of 2003 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, pp.286-295 (2003)
- [Zaki 02a] Zaki, M. J. and Hsiao, C.-J.: CHARM: An Efficient Algorithm for Closed Itemset Mining, *Proc. of 2nd SIAM International Conference on Data Mining*, pp.457-473 (2002)
- [Zaki 02b] Zaki, M. J.: Efficiently Mining Frequent Trees in a Forest, *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.71-80 (2002)
- [Zaki 06] Zaki, M. J. and Aggarwal, C. C.: XRules: An Effective Structural Classifier for XML Data, *Machine Learning Journal*, special issue on Statistical Relational Learning and Multi-Relational Data Mining, Vol. 62, No. 1-2, pp.137-170 (2006)
- [Zhu 07] Zhu, F., Yan, X., Han, J. and Yu, P. S.: gPrune: A Constraint Pushing Framework for Graph Pattern Mining, *Proc. of 2007 Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'07)*, LNAI 4426, pp.388-400 (2007)
- [Zou 06] Zou, L., Lu, Y., Zhang, H. and Hu, R.: Mining Frequent Induced Subtree Patterns with Subtree-Constraint, *Workshops Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006)*, pp.3-7 (2006)

〔担当委員：森田 千絵〕

2007 年 6 月 11 日 受理

—— 著 者 紹 介 ——

尾崎 知伸(正会員)

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。2001 年同研究科博士課程単位取得退学。2002 年より同研究科講師。2005 年より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などの研究に従事。

大川 剛直(正会員)

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などの研究に従事。IEEE、情報処理学会、電子情報通信学会、電気学会などの会員。