A-011

線形回数の所属性質問と 1 つの正例による 無順序木パターン言語族に対する質問学習アルゴリズム
A Learning Algorithm for Unordered Tree Pattern Languages Using One Positive Example and a Linear Number of Membership Queries

石灘 洸樹 [1] 正代 隆義 [2] 松本 哲志 [3] 内田 智之 [4]
Hiroki Ishinada Takayoshi Shoudai Satoshi Matsumoto Tomoyuki Uchida

1. はじめに

計算論的学習理論において，Angluin[1]によって提案された質問学習モデルは，質問を用いた学習の数学的モデルである．このモデルは，データベースから特徴的なパターンを抽出するためのデータマイニング技術としても応用可能である．

質問学習モデルは，特に所属性質問や等価性質問を通じて，教師からのフィードバックを基に学習を進める．この質問学習モデルでは，文字列パターンに関する理論的結果が多く得られている[2][3]．例えば，正則パターン言語のクラスは，1 つの正例，すなわちパターンにマッチする文字列から多項式回数の所属性質問を用いて同定可能であることが知られている．ここで所属性質問とは，文字列またはグラフを入力として，それが特定の言語またはグラフ言語の要素であるか否かを問う質問のことである．

Matsumoto ら[4]は，1 つの正例とその長さNの$O(N)$回の所属性質問を用いて正則パターン言語のクラスを同定する質問学習アルゴリズムを与えた．Uchida ら[5]は，ある種のグラフ文法により生成可能な順序木構造パターン言語の質問学習可能性を詳細に議論した．これらの結果から，無順序木構造パターン言語のクラスに関しては，互いに異なる変数ラベルを持つ無順序木構造パターン言語のクラスが，1 つの正例から多項式回数の所属性質問を用いて質問学習可能であることがわかる．

石灘ら[6]は，無順序木構造データベースに対する二値分類問題を著しく高い精度で計算するグラフ畳み込みネットワーク（GCN と略す）を用いて，葉のみに変数を持つ無順序木構造パターンの GCN 協調質問学習モデルを提案し，無順序木構造パターンを発見する新しい手法とその有効性を示した．そのモデルでは，1 つの正例から多項式回数の所属性質問を用いた既存の質問学習アルゴリズムを用いている[7]．その質問学習アルゴリズムでは，頂点及び辺ラベルの数が有限個である場合，頂点数Nの正例に対して$O(N^2)$回の所属性質問を必要とする．

本論文では，頂点及び辺ラベルの数が有限個である場合でも，1 つの正例とその頂点数Nの$O(N)$回の所属性質問を用いて無順序木パターン言語のクラスを同定する質問学習アルゴリズムを与える．さらに，人工的に生成した無順序木データに対して所属性質問の回数を実測調査し，線形回数，特にその係数を確認したので，その結果を報告する．

1 福岡工業大学大学院工学研究科 Graduate School of Engineering, Fukuoka Institute of Technology
2 福岡工業大学情報工学部 Faculty of Information Engineering, Fukuoka Institute of Technology
3 東海大学理学部 Faculty of Science, Tokai University
4 広島市立大学大学院情報科学研究科 Graduate School of Information Sciences, Hiroshima City University

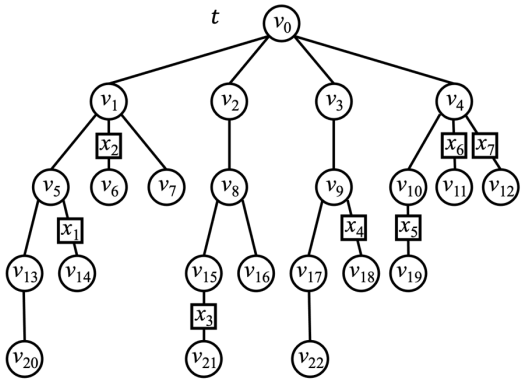

図 1: 線形無順序木パターンtにおいて，例えば，$[v_1, v_6]$は変数ラベルx_2を持つ変数である．

2. 準備

本論文では，サイクルを持たない連結なグラフを木と呼ぶ．さらに，根と呼ばれる頂点を 1 つ指定した木のことを無順序木と呼ぶ．ΣとΛを有限アルファベットとする．また，Xを$(\Sigma \cup \Lambda) \cap X = \emptyset$である無限アルファベットとする．頂点集合$V_T$と辺集合$E_T$を持つ無順序木を$T = (V_T, E_T)$とする．異なる2頂点$u, v \in V_T$が隣接するとき，$u$と$v$からなる辺を根からの距離が近い順に順序対で表す．辺$(u, v) \in E_T$に対して，uをvの親，vをuの子と呼ぶ．頂点vの親頂点を$p(v)$で表す．根でない次数1の頂点を葉と呼ぶ．無順序木は，同じ親を持つ頂点間に順序関係(兄弟関係)を持たない．長さkのチェーン(長さkの道だけからなる木，$k \geq 1$)に対して，次数 1 の頂点を根と定めた無順序木をP_kで表す．

頂点集合V_Tの各頂点には，Σに属す記号が，辺集合E_Tの各辺にはΛに属す記号がラベル付けされているとする．Σに属す記号を頂点ラベル，Λに属す記号を辺ラベルと呼ぶ．Xに属す記号を変数ラベルと呼ぶ．2 つの無順序木T_1とT_2に対して，T_1の根をT_2の根に写し，頂点ラベルと辺ラベルを保存するグラフ同型写像が存在するとき，$T_1 \equiv T_2$と記す．以下では，空語をεとして，$\Sigma = \{\varepsilon\}$，$\Lambda = \{\varepsilon\}$とする．本論文で提案するアルゴリズムを任意の頂点ラベル集合Σと辺ラベル集合Λへ一般化することは容易である．

2.1 無順序項木パターンと無順序木パターン

本論文では，2 次元の変数を持つ無順序木パターンのみを扱う．3 次元以上の変数とそれを持つ無順序項木パターンについては参考文献[8]に詳細な定義がある．

$T = (V_T, E_T)$を無順序木とする．H_TをE_Tの部分集合とする．**TとH_Tから得られる無順序項木パターン**とは，次の条件(1)-(4)を満たす 3 つ組$t = (V_t, E_t, H_t)$である：(1) $V_t = V_T$，(2) $E_t = E_T \setminus H_T$，(3) $H_t = \{[u, v] \mid (u, v) \in H_T\}$，(4) V_tに属

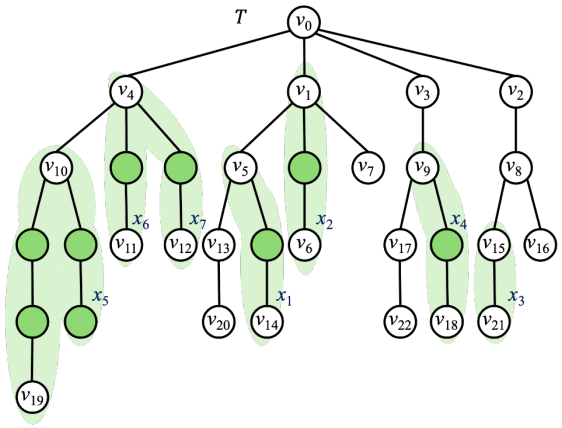

図 2: 無順序木 T は，図 1 の線形無順序木パターンtに図 3 の代入θを適用して得られる$t\theta$である．

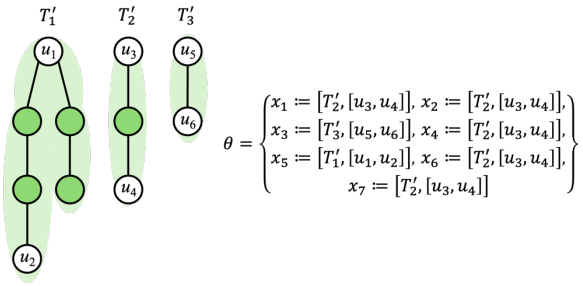

図 3: 代入θは変数ラベル $x_1, \ldots, x_7$ の束縛を含む集合である．

す頂点の頂点ラベル，E_tに属す辺の辺ラベルは，それぞれに対応するV_Tの頂点ラベル，E_Tの辺ラベルと同じである．H_tの要素を変数と呼び，H_tを変数集合と呼ぶ．変数集合H_tの各変数にはXに属す記号がラベル付けされているとする．3 つ組$t = (V_t, E_t, H_t)$が提示されれば，それを作成するために使われる無順序木$T = (V_T, E_T)$とE_Tの部分集合H_Tは一意に定まるので，$t = (V_t, E_t, H_t)$を単に**無順序項木パターン**と呼ぶ．H_Tに属すすべての辺(u, v)の子vがTの葉であるならば，TとH_Tから得られる無順序項木パターンを，**無順序木パターン**と呼ぶ．図 1 に無順序木パターンの例をあげる．本論文では，変数と変数ラベルを図 1 のように四角で表す．四角に囲まれた文字は変数ラベルを表し，その変数が含む頂点を線で結ぶ．

無順序項木パターン$t = (V_t, E_t, H_t)$の変数集合H_tの全ての変数が互いに異なる変数ラベルを持つとき，tを線形無順序項木パターンと呼ぶ．線形無順序項木パターンtが無順序木パターンであれば，tを**線形無順序木パターン**と呼ぶ．図 1 の無順序木パターンは，線形無順序木パターンである．

2.2 束縛と代入

T'を2個以上の頂点を持つ無順序木とする．u_0をT'の根，u_1をT'の根以外の頂点とし，$\sigma = [u_0, u_1]$とする．xをXに属す変数ラベルとする．このとき，$x := [T', \sigma]$という形を**束縛**という．$t = (V_t, E_t, H_t)$を無順序項木パターンとする．変数ラベルxを持つ変数がH_tの要素であるとき，その変数を$h_1, \ldots, h_k (k \geq 0)$とする．束縛$x := [T', \sigma]$を変数$h_1, \ldots, h_k$に次のように同時に適用して得られる無順序項木パターンを$t\{x := [T', \sigma]\}$と書く:

(1) $T'_1, \ldots, T'_k$を無順序木T'のコピーとする．

(2) 変数$h_j = \left[v_0^{(j)}, v_1^{(j)}\right] (1 \leq j \leq k)$に関して，$H_t$から変数$h_j$を削除し，$T'$の頂点$u_0, u_1$に対応する$T'_j$の頂点$u_0^{(j)}, u_1^{(j)}$を，この順序で変数$h_j$の頂点$v_0^{(j)}, v_1^{(j)}$と同一視する．同一視後の頂点ラベルは，頂点$v_0^{(j)}, v_1^{(j)}$の頂点ラベルを継承する．

代入θとは，$x_i\ (1 \leq i \leq n)$がXの異なる変数ラベルである束縛の有限集合$\{x_1 := [T'_1, \sigma_1], \ldots, x_n := [T'_n, \sigma_n]\}$である．ただし，$T'_1, \ldots, T'_n$には変数ラベル$x_1, \ldots, x_n$を持つ変数が現れないと仮定する．代入$\theta$による$t$のインスタンスとは，$t$に対して，$\theta$の全ての束縛を適用して得られる無順序項木パターンである．代入θによるtのインスタンスを$t\theta$と書く．無順序項木パターンtと無順序木Tに対して，代入θが存在して$t\theta \equiv T$となるとき，tはTとマッチするという．

2.3 線形無順序木パターン言語

Σ, Λ, Xを頂点ラベル集合，辺ラベル集合，変数ラベル集合とする無順序項木パターンの全体を$\mathcal{UTP}_{\Sigma,\Lambda,X}$とする．$\mathcal{UTP}_{\Sigma,\Lambda,X}$に属す線形無順序木パターンの全体を$\mathcal{LUTP}_{\Sigma,\Lambda,X}$とする．また，$\Sigma, \Lambda$を頂点ラベル集合，辺ラベル集合とする無順序木の全体を$\mathcal{UT}_{\Sigma,\Lambda}$とする．以降，無順序木$T = (V_T, E_T)$を，変数を持たない無順序項木パターン$T = (V_T, E_T, \emptyset)$とみなす．無順序項木パターン$t \in \mathcal{UTP}_{\Sigma,\Lambda,X}$に対して，$t$の無順序項木パターン言語$L(t) \subseteq \mathcal{UT}_{\Sigma,\Lambda}$を次のように定義する: $L(t) = \{t\theta \in \mathcal{UT}_{\Sigma,\Lambda} \mid \theta$は代入$\}$．図 1 の$t$に対して，図 3 に表された代入を適用して得られる無順序木Tを図 2 にあげる．

本論文では，線形無順序木パターンしか扱わない．以降，線形無順序木パターンを単に無順序木パターンと呼ぶ．

3. Angluin の質問学習モデル

Angluin によって提案された質問学習モデルでは，学習対象に関する質問に対して，常に正答を返す完璧な教師であるオラクルを仮定して質問回数に関する解析を行う．

学習目標である無順序木パターン$t_* \in \mathcal{LUTP}_{\Sigma,\Lambda,X}$に対して，無順序木$T \in \mathcal{UT}_{\Sigma,\Lambda}$が$T \in L(t_*)$を満たすとき$T$を$t_*$の正例と呼び，そうでないときを負例と呼ぶ． $t_* \in \mathcal{LUTP}_{\Sigma,\Lambda,X}$に対する**所属性質問**とは，任意の無順序木$T \in \mathcal{UT}_{\Sigma,\Lambda}$を入力とし，$T$が正例であるとき$Yes$を，負例であるとき$No$を回答する質問である． $t_* \in \mathcal{LUTP}_{\Sigma,\Lambda,X}$の所属性質問に回答するオラクルは，$t_*$の情報($t_*$の表現)を何ら示すことなく，所属性質問に回答するアルゴリズムである．

t_*のオラクルを$\mathcal{O}(t_*)$と記す．無順序木$T \in \mathcal{UT}_{\Sigma,\Lambda}$に対する$t_*$のオラクル$\mathcal{O}(t_*)$の回答を$\mathcal{O}(t_*)(T)$と記す．すなわち，オラクル$\mathcal{O}(t_*)$は，$T \in L(t_*)$を満たすとき$\mathcal{O}(t_*)(T) = Yes$，そうでないとき$\mathcal{O}(t_*)(T) = No$を返す．無順序木パターン$t_* \in$

表 1: 線形無順序木パターン質問学習アルゴリズム

アルゴリズム$\mathcal{LUTP}_{\Sigma,\Lambda,X}$_$Query$;
入力: $t_* \in \mathcal{LUTP}_{\Sigma,\Lambda,X}$の正例$T \in L(t_*)$;
仮定: $t_* \in \mathcal{LUTP}_{\Sigma,\Lambda,X}$に対する所属性質問に答えるオラクル$\mathcal{O}(t_*)$;
Step 1. 変数の数を特定する;
Step 2. 最小正例を求める;
Step 3. 変数を特定する;
出力: 無順序木パターン$t_* \in \mathcal{LUTP}_{\Sigma,\Lambda,X}$;

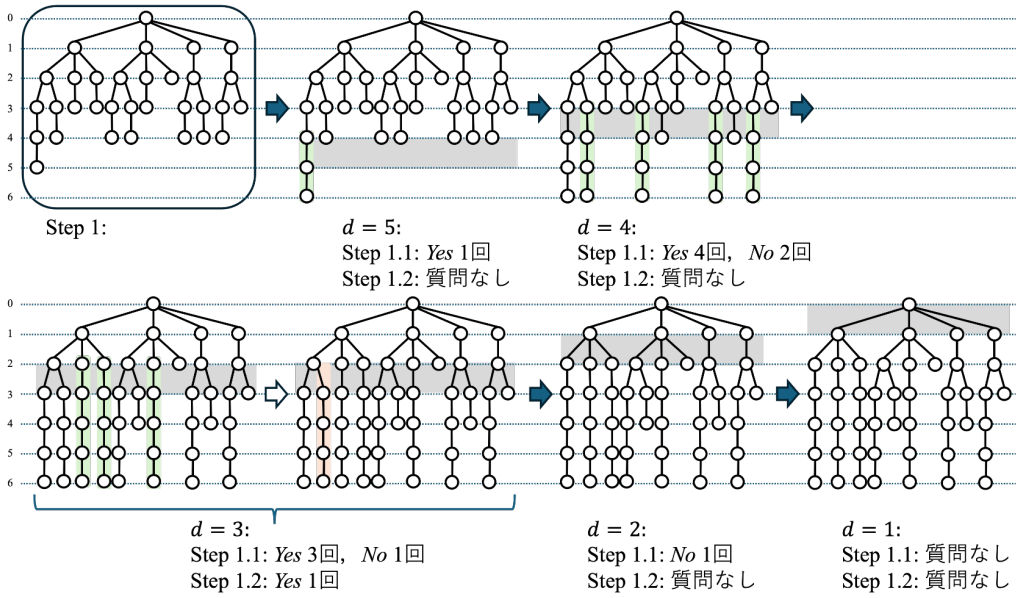

図 4: アルゴリズム$\mathcal{LLUTT}_{\Sigma,\Lambda,X}_Query$の Step 1 の実行例

$\mathcal{LUTP}_{\Sigma,\Lambda,X}$ と無順序木 $T \in \mathcal{UT}_{\Sigma,\Lambda}$ に対して，$\Theta_{t_*}(T) = \{\theta \mid t_*\theta \equiv T$，ただし$\theta$は$t_*$への代入$\}$と定める．言い換えれば，オラクル$\mathcal{O}(t_*)$は，$|\Theta_{t_*}(T)| > 0$を満たすとき$\mathcal{O}(t_*)(T) = Yes$を返し，$|\Theta_{t_*}(T)| = 0$とき$\mathcal{O}(t_*)(T) = No$を返すアルゴリズムである．

本論文では，質問学習アルゴリズム$\mathcal{LUTP}_{\Sigma,\Lambda,X}_Query$ (表 1)を提案することで，次の定理を示す．

定理 無順序木パターン$t_* \in \mathcal{LUTP}_{\Sigma,\Lambda,X}$に対する所属性質問に答えるオラクル$\mathcal{O}(t_*)$を仮定する．頂点数$N$ $(N \geq 5)$の正例$T \in L(t_*)$が与えられたとき，$\mathcal{O}(t_*)$への高々$4N$回の所属性質問により，t_*の表現，すなわち$t = (V_t, E_t, H_t)$を同定可能である．

4. 線形無順序木パターンの質問学習アルゴリズム

本章では，オラクル$\mathcal{O}(t_*)$を用いた質問学習アルゴリズム$\mathcal{LUTP}_{\Sigma,\Lambda,X}_Query$ (表 1)の 3 ステップを詳細に示す．正例Tの高さ(深さの最大値)をhとする．Tの頂点vを根とするTの部分無順序木を$T[v]$と記述する．同様に，t_*の頂点v'を根とするt_*の部分無順序木パターンを$t_*[v']$と記述する．

Step 1. 変数の数を特定する;

1. $d := h$;
2. 深さdの全ての葉vに対して，次を行う:
   辺$(p(v), v)$を変数$[p(v), v]$とみなし，P_{h-d+1}を束縛して，無順序木T'を作成する;
   if $\mathcal{O}(t_*)(T') = Yes$ **then** $T := T'$;
3. 深さ$d-1$にある全頂点vに対して，次を行う:
   vの子を$c_1, \ldots, c_n$ $(n \geq 2)$とする;
   $C := \{c_i \mid T[c_i] \equiv P_{h-d+1}$ $(1 \leq i \leq n)\}$;
   for c **in** C **do**
   if $|C| > 1$ **then**
   Cからcを削除する;
   Tから$T[c]$を削除して，無順序木T'を作成する．ただし，cと$(p(c), c)$も一緒に削除する;
   if $\mathcal{O}(t_*)(T') = Yes$ **then** $T := T'$;
4. **if** $d > 1$ **then** $d := d - 1$; **goto** 2 **else output** T;

図 4 に，図 1 の無順序木パターンをt_*とし，図 2 の無順序木をT とする Step 1 の実行例を示す．Step 1 では高々$2N$回の所属性質問が実行される．もし$\mathcal{O}(t_*)(T') = Yes$ならば，$|\Theta_{t_*}(T)| \geq |\Theta_{t_*}(T')| > 0$であることに注意する．

補題 1 アルゴリズム$\mathcal{LUTP}_{\Sigma,\Lambda,X}_Query$の入力を$T \in L(t_*)$とする．このとき Step 1 は深さ$h+1$の葉の数が$t_*$の変数の数と等しい$t_*$の正例を出力する．

Step 2. 最小正例を求める;

1. $d := 1$;
2. 深さdの全ての頂点vに対して，次を行う:
   if $T[v] \equiv P_{h-d+1}$ **then**
   Tから$T[v]$を削除して，無順序木T'を作成する．ただし，vと$(p(v), v)$は削除しない;
   if $\mathcal{O}(t_*)(T') = Yes$ **then** $T := T'$;
3. **if** $d < h$ **then** $d := d + 1$; **goto** 2 **else output** T;

図 5 に，図 4 の続きとして，Step 2 の実行例を示す．Step 2 では高々N回の所属性質問が実行される．

表 2: 変数の位置を特定する手続き Find_Variable

procedure Find_Variable(T, d):
入力: 無順序木T, 深さd;
if Tの根が子を持つ **then**
根の子を$c_1, \ldots, c_n$ $(n \geq 2)$とする．
ただし，$T[c_1], \ldots, T[c_n]$は深さ列の降順である;
for c **in** $[c_1, \ldots, c_n]$ **do**
if $T[c]$が長さ 0 以上のチェーン **then**
$T[c]$の葉をℓとする;
$T[c]$をP_{h-d}に置換して無順序木T'を作成する;
if $\mathcal{O}(t_*)(T') = Yes$ **then**
$T := T'$;
辺$(p(\ell), \ell)$を「変数」とマークする;
else // $T[c]$がチェーンではない．
Find_Variable$(T[c], d+1)$;

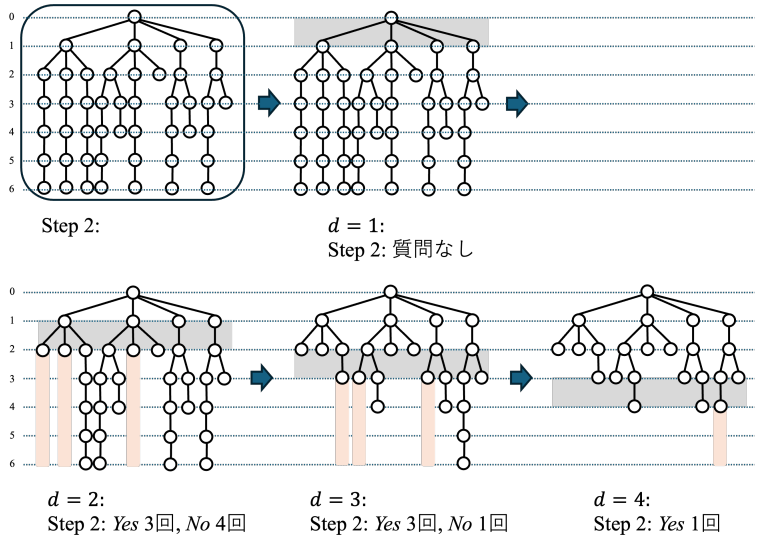

図 5: アルゴリズム$\mathcal{LUTP}_{\Sigma,\Lambda,X}$_$Query$の Step 2 の実行例

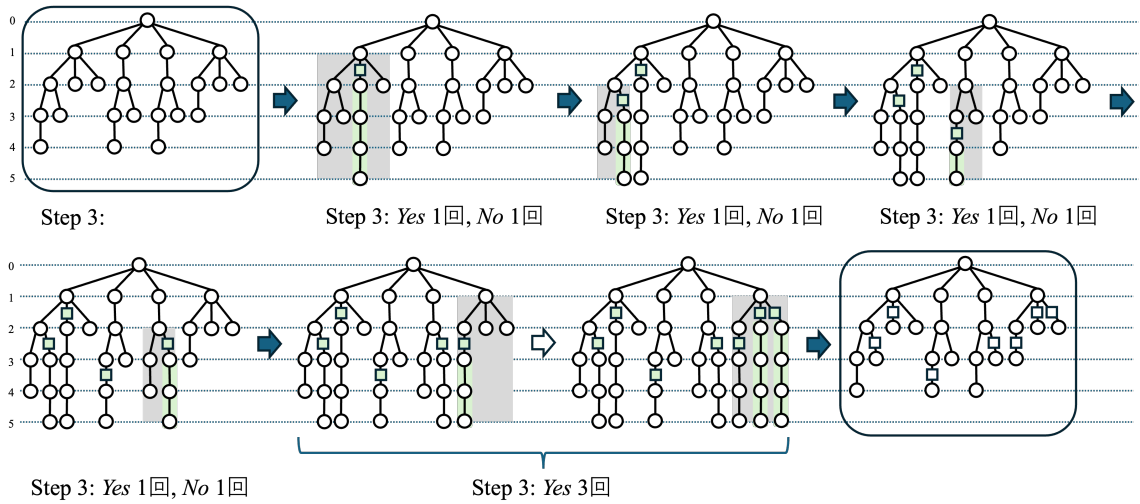

図 6: アルゴリズム$\mathcal{LUTP}_{\Sigma,\Lambda,X}$_$Query$の Step 3 の実行例

補題 2 アルゴリズム$\mathcal{LUTP}_{\Sigma,\Lambda,X}$_$Query$における Step 1 の出力を$T \in L(t_*)$とする．このとき Step 2 は$t_*$の変数を辺に置き換えて得られる$t_*$の正例（最小正例）を出力する．

Step 3. 変数を特定する;

順序木において，左側の子を優先して深さ優先探索して，訪れた頂点の深さを並べた整数列のことを深さ列と呼ぶ．無順序木では，子の順序を交換して得られる全ての順序木の深さ列において，辞書式順序で最大の深さ列を持つ順序木を左深さ優先木と呼ぶ．無順序木では，子の順序を交換して得られる全ての順序木の深さ列において，辞書式順序で最大の深さ列を持つ順序木を左深さ優先木と呼ぶ．

Step 3 では，Tを左深さ優先木に変換して，手続き Find_Variable (表 2)に対し，Find_Variable$(T, 0)$を実行する．

図 6 に，図 5 の続きとして，Step 3 の実行例を示す．Step 3 では高々N回の所属性質問が実行される．

補題 3 アルゴリズム$\mathcal{LUTP}_{\Sigma,\Lambda,X}$_$Query$における Step 2 の出力を$T \in L(t_*)$とする．このとき Step 3 は$t_*$を出力する．

Step 3 において，変数の位置が正しく特定できることは，Step 3 への入力となる無順序木Tの深さdの頂点vの子$c_1, \ldots, c_n$ $(n \geq 2)$を根とする部分木$T[c_1], \ldots, T[c_n]$が深さ列の降順に Find_Variable が実行されることにより保証される．以下，証明の概略を述べる．証明は Find_Variable が訪れた頂点順の帰納法による．Step 2 の出力Tが最小正例であるから，t_*の変数を辺で置き換えた無順序木はTと同型である．頂点vの子$c_1, \ldots, c_n$ $(n \geq 2)$に対応するt_*の頂点を$c'_1, \ldots, c'_n$ $(n \geq 2)$とする．頂点vの部分木$T[c_i]$ $(1 \leq i \leq n)$において，Find_Variable$(T[c_i], d+1)$が実行されているとき，$k > i$に対して，$T[c_k]$が$T[c_i]$と異なる深さ列を持つならば，$t_*[c'_i]$が$T[c_k]$にマッチすることはない．なぜなら，無順序木パターンが自身の深さ列より小さい無順序木にマッチすることはないからである．$T[c_k]$が$T[c_i]$と同じ深さ列を持つならば，$T[c_i]$の処理における$t_*[c'_i]$の変数位置は，$T[c_k]$に対応する$t_*[c'_k]$の変数位置より深さ優先で特定される．一方，$k < i$では，すでに Find_Variable$(T[c_k], d+1)$の実行は終了しているので，$T[c_k]$における$t_*[c'_k]$に対応する変数の位置は特定されている．もし，Find_Variable$(T[c_i], d+1)$を実行している途中の$T[c_i]$に$t_*[c'_k]$がマッチするのであれば，すでに変数位置が特定されている頂点vの部分木$T[c_{k'}]$ $(k' < i)$が存在して，$t_*[c'_i]$は$T[c_{k'}]$にマッチしなければならない．

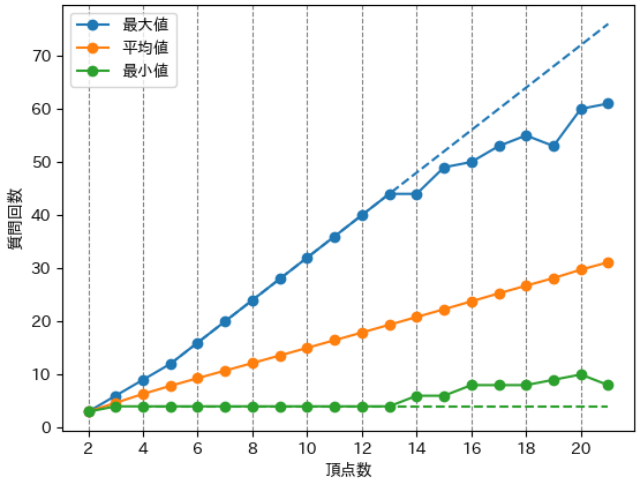

図 7: 入力が最小正例とするときの質問回数の推移

これらの事実から，Find_Variable($T[c_i], d+1$)が$T[c_i]$の変数位置を正確に特定することが示される．

以上の Step 1, 2, 3 により，定理の所属性質問の回数が示される．また補題 1, 2, 3 より，アルゴリズムの正当性が示される．各補題の詳細な証明は略す．

5. 実験と考察

本章では，人工データを用いて行なった 2 つの計算機実験による解析結果を報告する．本実験では無順序木パターンの所属性質問を行う際のオラクルとして，Shoudai ら[8]によって提案された線形無順序項木パターンの多項式時間マッチングアルゴリズムを使用した．このアルゴリズムは，学習目標である無順序木パターン$t_* \in \mathcal{LUTP}_{\Sigma,\Lambda,X}$から照合のためのルールを作成して無順序木$T$の頂点と$t_*$の頂点の間の対応関係を求める．それにより$t_*$が$T$とマッチするか否かを判定する．以降，葉に接続する辺または変数を葉辺と呼ぶ．

5.1 最小正例から同定する際の質問回数の解析結果

アルゴリズム$\mathcal{LUTP}_{\Sigma,\Lambda,X}$_$Query$への入力を学習目標である$t_* \in \mathcal{LUTP}_{\Sigma,\Lambda,X}$の頂点数最小の正例とする．オラクル$\mathcal{O}(t_*)$を用いた質問学習アルゴリズム$\mathcal{LUTP}_{\Sigma,\Lambda,X}$_$Query$ (表 1)の入力をt_*の変数を辺とみなした無順序木T_{t_*}とし，各 Step でオラクルに所属性質問を行う回数を求めた．ここでは，あらかじめ定めた頂点数を持つ全ての無順序木を解析対象とするために，無順序木の列挙アルゴリズム[9]を使用し，頂点数21まで全無順序木を生成した．生成した各無順序木の葉辺を変数に置き換えることによって，全ての無順序木パターンを作成し，頂点数と所属性質問を行う回数に関する解析を行った．ただし，頂点数 2 から 13 は全ての無順序木パターンに対して解析を行なったが，計算時間が非常に大きくなる頂点数 14 以降は，頂点数 13 の無順序木パターンの数 409963 個を参考に 410000 個の無順序木パターンをランダムに生成し，解析を行った．

実験結果の折れ線グラフを図 7 に，その数値結果を表 3 に示す．横軸に頂点数，縦軸に質問回数を表す．各頂点の質問回数に対して最大値を青色，最小値を緑色，平均値を橙色でプロットし，折れ線グラフで表現する．いずれも定数倍で増加していることが確認できる．全ての無順序木パターンを解析した頂点数 13 までの結果から，頂点数を

表 3: 図 7 の数値結果

頂点数	木の数	最大	最小	平均
2	1	3	3	3.00
3	3	6	4	4.67
4	9	9	4	6.33
5	28	12	4	7.86
6	88	16	4	9.31
7	284	20	4	10.73
8	927	24	4	12.15
9	3074	28	4	13.57
10	10300	32	4	15.01
11	34880	36	4	16.45
12	119109	40	4	17.90
13	409963	44	4	19.35
14	410000	44	6	20.82
15	410000	49	6	22.28
16	410000	50	8	23.77
17	410000	53	8	25.27
18	410000	55	8	26.72
19	410000	53	9	28.14
20	410000	60	10	29.73
21	410000	61	8	31.12

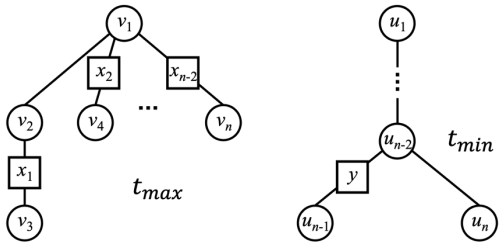

図 8: 頂点数が同じとき質問回数が最大となる無順序木パターンt_{max}と最小となる無順序木パターンt_{min}

N ($N \geq 5$)とするとき，質問回数の最大値が$4(N-2)$ (図 7: 青破線)，最小値が4 (図 7: 緑破線)であることが確認できた．

実験により，質問回数が最大となる無順序木パターンt_{max}は，$N \geq 5$の場合，葉辺が変数である長さ 2 のチェーンが 1 つと葉辺に変数を持つ長さ 1 のチェーンが 2 以上ある場合である(図 8 (左))ことが確認できた．これは根と$N-1$個の葉が接続された無順序木パターンに比べ，Step 2 で深さ 1 または 2 のどちらが変数であるか検証する必要があるため質問回数が多くなる．$N = 2,3,4$の場合の最大質問回数はそれぞれ3,6,9である．また，質問回数が最小となる無順序木パターン($N \geq 3$)は図 8 (右)の無順序木パターンt_{min}である．図ではt_{min}の頂点集合を$\{u_1, u_2, \ldots, u_n\}$ ($n = N$, $N \geq 3$)としている．根u_1から頂点u_{n-2}まで分岐を持たない頂点が接続されており，u_{n-2}に接続する長さ 1 のチェーンで葉辺が変数であるものが 1 つ，葉辺が辺であるものが 1 つ持つ場合である．この無順序木パターンは，頂点数Nのチェーンである無順序木パターンに比べ，Step 2 でのチェーンを短くするための質問を行う必要がない．そのため，質問回数が最小となる．

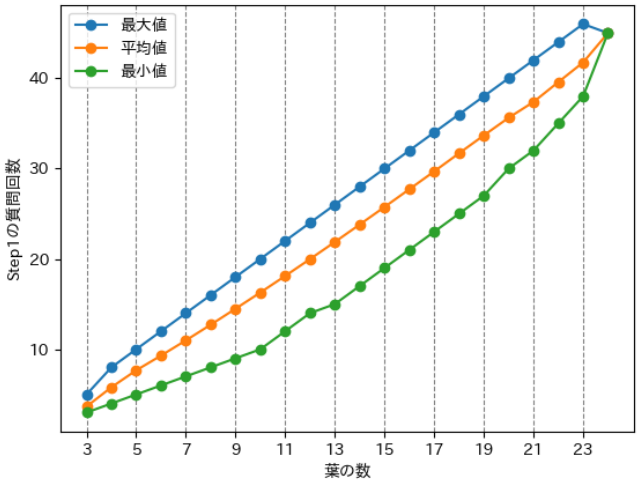

図 9: 葉の数と質問回数の関係

5.2 正例の葉の数に関する質問回数の解析結果

アルゴリズム$\mathcal{LUTP}_{\Sigma,\Lambda,X}$-$Query$への入力の頂点数を，学習目標である$t_* \in \mathcal{LUTP}_{\Sigma,\Lambda,X}$の頂点数の 1 倍から 2 倍まで変化させて，正例の葉の数に関する質問回数の解析を行なった．5.1 章において全ての無順序木パターンを調査することができた頂点数 13 を対象とし，頂点数 13 から成る無順序木パターンt_*に対して，正例となる頂点数 26 の無順序木をランダムに 100 個生成した．生成した正例の集合をS_+とする．S_+に属す無順序木を$\mathcal{LUTP}_{\Sigma,\Lambda,X}$-$Query$の入力として，Step 1 まで行う．これをランダムに 40000 回繰り返した．異なる入力(無順序木)であっても学習目標である無順序木パターンt_*が同じであれば，無順序木の頂点数に関わらず Step 2 と Step 3 の質問回数は変わらないので，本実験は Step 1 まで実行し，その結果を解析した．

実験結果の折れ線グラフを図 9 に，その数値結果を表 4 に示す．入力となる無順序木の頂点数を固定した場合，質問回数は無順序木の葉の数に比例して増加することがわかる．質問回数は，無順序木の頂点数に関わらず，葉の数のみに依存する．また，Step 1 までの質問回数は葉の数の 2 倍以下であることがわかる．これは，葉にチェーン型を代入するステージと分岐を減らすステージを同じ深さで行っているからで，定理と同じ結果になることが確認された．

6. おわりに

本論文では新たなアルゴリズムとして，一つの正例と無順序木パターンを線形回数での所属性質問で同定可能なアルゴリズムを提案し，理論的かつ実験的に正当性を確認した．そこでは質問回数は高々$4N$ (Nは無順序木の頂点数)になることが確認できた．また，処理 Step 1 は，入力とする正例の葉の数に依存し，Step 2 は，無順序木パターンの変数の位置に依存し，Step 3 は無順序木パターンの葉の数に依存することが確認できた．

今後の課題は，平均解析が線形時間となることを理論的に解析することである．さらに，グラフ畳み込みネットワーク (GCN)をオラクルとして，1 つの正例と線形回数の所属性質問によって無順序木パターンを同定する本論文提案の質問学習アルゴリズムを適用し，従来手法との計算時間と精度の比較・検討を行うことである．

表 4: 図 9 の数値結果

葉の数	木の数	最大値	最小値	平均値
3	3	5	3	3.67
4	49	8	4	5.80
5	676	10	5	7.67
6	5990	12	6	9.30
7	33108	14	7	10.99
8	127383	16	8	12.72
9	355145	18	9	14.49
10	739542	20	10	16.28
11	1182214	22	12	18.12
12	1477072	24	14	19.98
13	1471439	26	15	21.88
14	1178279	28	17	23.81
15	765411	30	19	25.75
16	404812	32	21	27.71
17	175357	34	23	29.68
18	61441	36	25	31.69
19	17343	38	27	33.67
20	3966	40	30	35.62
21	682	42	32	37.37
22	79	44	35	39.57
23	8	46	38	41.75
24	1	45	45	45.00

謝辞

本研究は JSPS 科研費 JP20K04973，JP21K12021，JP24K15090 の助成を受けたものです．

参考文献

[1] Angluin, D.: Queries and Concept Learning, Matching Learning, Vol.2, No.4, pp.319–342 (1988).
[2] Angluin, D.: Finding patterns common to a set of strings, Journal of Computer and System Sciences, Vol.21, No.1, pp.46-62 (1980).
[3] Angluin, D.: Learning Regular Sets from Queries and Counterexamples, Information and Computation, Vol.75, No.2, pp.87-106 (1987).
[4] Matsumoto, S., Uchida, T., Shoudai, T., Suzuki, Y., and Miyahara, T.: An Efficient Learning Algorithm for Regular Pattern Languages Using One Positive Example and a Linear Number of Membership Queries, IEICE Trans. Inf. & Syst., Vol.E103-D, No.3, pp.526-539 (2020).
[5] Uchida, T., Matsumoto, S., Shoudai, T., Suzuki, Y., and Miyahara, T.: Exact Learning of Primitive Formal Systems Defining Labeled Ordered Tree Languages via Queries, IEICE Trans. Inf. & Syst., Vol.E102-D, No.3, pp.470-482 (2019).
[6] 石灘 洸樹，正代 隆義，内田 智之，松本 哲志：無順序木パターンに対する高精度グラフ畳み込みネットワークをオラクルとする質問学習モデルの解析，研究報告数理モデル化と問題解決 (MPS), 2023-MPS-143, 15, pp.1-8 (2023).
[7] Amoth, T.R. Cull, P., and Tadepalli, P.: On Exact Learning of Unordered Tree Patterns, Machine Learning, Vol.44, pp.211-243, (2001).
[8] Shoudai, T., Miyahara, T., Uchida, T., Matsumoto, S., and Suzuki, Y.: An Efficient Pattern Matching Algorithm for Unordered Term Tree Patterns of Bounded Dimension, IEICE Trans. Fundamentals, Vol.E101-A, No.9, pp.1344-1354 (2018).
[9] Nakano, S. and Uno, T.: Constant Time Generation of Trees with Specified Diameter, Graph-Theoretic Concepts in Computer Science (WG2004), LNCS 3353, pp.33-45 (2004).