4L-04

バンディットアルゴリズムを用いた近似的最大共通部分木の計算

森 光太朗 †　　山本章博 ‡　　吉仲亮 ‡
京都大学工学部情報学科 †　　京都大学情報学研究科 ‡

1 はじめに

XML 文書や HTML 文書などの半構造データからのデータマイニングや機械学習では，データをラベル付き順序木として扱うため，2 つの木の類似度を計算する必要がある．最大共通部分木はそれらの類似度を表す指標として広く使われている．最大共通部分木はマッピングによって定義され，Tai マッピングはその 1 つである．現在提案されている木と木の Tai マッピングに基づく最大共通部分木を求めるための動的計画法を用いたアルゴリズムは，真の最大共通部分木を求める．しかし，入力の木のサイズ n に対して $O(n^3)$ の時間と $O(n^2)$ のメモリ量が必要となり，巨大な木に対しては適用できないことがある．そこで，本研究においては，2 つの巨大な木が与えられたときに，その Tai マッピングに基づく共通部分木のなかでなるべくそのノード数が大きいものを求める手法を提案する．

2 準備

2.1 Tai マッピングに基づく最大共通部分木問題

以下の三つの操作を木 T への編集操作という．

1. **置換**: ノード v のラベル $l(v)$ を l' に置き換える．
2. **挿入**: T のノード v を新たなノード v' の親，v の子を v' の子となるように枝を付け替え，v' を T のノード集合に追加する．
3. **削除**: ノード v の親を v の子の親となるように枝を付け替え，v を T のノード集合から削除する．

これらの編集操作についてラベルに注目し，置換を $l \to l'$,λ を空白記号として挿入を $\lambda \to l$, 削除を $l \to \lambda$ と表す. そして，以下のようにコストを与える．

$$\text{置換のコスト：} \quad \gamma(l \to l') = \begin{cases} 0 & \text{if } l = l', \\ 1 & \text{if } l \neq l'. \end{cases}$$

$$\text{挿入のコスト：} \quad \gamma(\lambda \to l) = 1.$$

$$\text{削除のコスト：} \quad \gamma(l \to \lambda) = 1.$$

ノード v_1, v_2 は木 T_1 中のノード, ノード w_1, w_2 は木 T_2 であるとする. 集合 $M \subseteq V(T_1) \times V(T_2)$ は, 任意の $(v_1, w_1), (v_2, w_2) \in M$ が以下の条件を満たすとき, 木 T_1 から木 T_2 への Tai マッピングという [1].

1. $v_1 = v_2 \iff w_1 = w_2$.　(1 対 1 の関係)
2. $v_1 < v_2 \iff w_1 < w_2$.　(先祖子孫の関係)
3. $v_1 \prec v_2 \iff w_1 \prec w_2$.　(左右の関係)

ここで, ノード v, w に対して，そのノードのラベルをそれぞれ $l(v), l(w)$ とし，I および J をマッピング M の要素ではない T_1 の頂点集合および T_2 の頂点集合とすると，Tai マッピングに対するコストが以下のように定義できる．

$$\sum_{(v,w)\in M} \gamma(l(v) \to l(w)) + \sum_{v\in I} \gamma(l(v) \to \lambda) + \sum_{w\in J} \gamma(\lambda \to l(w))$$

このコストが最小になるマッピングによって導かれる部分木を最大共通部分木という．

2.2 バンディットアルゴリズム

バンディットアルゴリズムとは多腕バンディット問題を解くアルゴリズムの総称である. 多腕バンディット問題とは，k 台のスロットマシンがあり，そのスロットマシンにはそれぞれマシン毎に異なる確率分布の報酬を返す．その報酬の確率分布を知らないプレイヤーはそのマシンを 1 回プレイするごとにその報酬を得ることができる．プレイヤーが選択するマシンは 1 回毎に変更しても良い. この問題の目的は, 有限回数の試行でプレイヤーが得られる報酬の合計をできるだけ大きくすることである. 代表的なアルゴリズムとして UCB1[2] 等がある．

3 提案手法

この提案手法は，2 つのラベル付き根付き順序木が与えられたとき，その Tai マッピングに基づく共通部分木の中でなるべく大きなものを取ってくることを目的に設計されている．

森 F_1 から適当にノード v を選び，v を基準に F_1 を分割統治していく．そしてもう一方の森から上記で選ばれたノードに合わせて，なるべく得られるマッピングのコストが小さくなるような分割の基準点を見つけていく．そのノードの探索をバンディットアルゴリズムによる探索によって行う. この探索は，明らかに悪い分割の基準点のペアの探索を極力行わなず，良さそうな点に探索を集中させることで効率の良い探索ができると考えられる．

森 F_1, F_2 が入力された時の処理の流れを以下に示す．

1. F_1 から任意にノード v を選ぶ．
2. もう片方の全てのノードをバンディットアルゴリズムにおける腕であると見なし，1 で選んだノー

Computing an Approximately Largest Common Subtree with a Bandit Algorithm
†Kotaro MORI　‡Akihiro YAMAMOTO ‡Ryo YOSHINAKA
†Kyoto University, Faculty of Engineering Undergraduate School of Information and Mathematical Science
‡Graduate School of Informatics Kyto University

ドとペアになると，最もコストの少ないマッピングが取れそうなノードを探索する．そして，この探索の結果，選ばれたノードを w とする．

3. 選ばれたノードのペア (v, w) をマッピング M に追加し，そして 1 で選ばれたノード v をもとに F_1 を，v の祖先のパス $P_1 = anc(v) - v$，部分木 $F'_1 = T(v)$，左側 G_{11}，右側 G_{12} に分割する．
さらに，2 で選ばれたノード w をもとに F_2 を，w の祖先のパス $P_2 = anc(w) - w$，部分木 $F'_2 = T(w)$，左側 G_{21}，右側 G_{22} に分割する．
4. 3 で得られた子孫 F'_1, F'_2，左側 G_{11}, G_{12}，右側 G_{21}, G_{22} に対し，再帰的に手法を適用する．
5. 1〜4 を繰り返し，十分小さな森にまで分割できたら動的計画法によって Tai マッピングを求め，それらをマッピング M に統合することによって，元の 2 つの入力された木に対する Tai マッピングを得る．

また，この手法における報酬とは「ノード v に対して，もう片方の木からこのノードをマッピングのペアに取ったとき，どのくらいのコストのマッピングが望めそうか」という指標をスコア化したものである．本研究における報酬取得の手順を以下に示す．

1. 報酬を取りたいノードを基準に分解してできた森のノードのラベルを，その右側同士，左側同士，部分木同士の組にして，ポストオーダー順に並べて，ラベルの列の組を得る．
2. 2 で得られたラベルの列の組を系列 $X = x_1, ..., x_m$, $Y = y_1, ..., y_n$ として，それぞれについて以下のように辿っていく．
   整数 $i = 1, j = 1, reward = 0$ とする．(i), (ii) を $i = m$ もしくは $j = n$ になるまで繰り返す．
   (i) x_i と y_j が同じラベルならば，$reward$ に 1 を加えて，i, j それぞれに 1 を加える．
   (ii) x_i と y_j が違うラベルならば，i に 1 を加える，j に 1 を加える，i, j それぞれに 1 を加えることのどれか 1 つをランダムに行う．
3. 得られた左側についての $reward$，右側についての $reward$，部分木についての $reward$ の総和をもとの森の大きさで正規化したものを報酬とする．

4 実験

行った実験は以下の 2 つである．

1. 真の最大共通部分木のノード数を 1 としたとき，どの程度のノード数の共通部分木が得られたか調べる．実験結果のグラフは図 1 であり，横軸は入力木のノード数を表している．
2. 提案手法との速度の比較をする既存アルゴリズムは Klein のアルゴリズム [3] を用いた．実験結果を図 2 に示す．図において横軸は入力木のノード数，

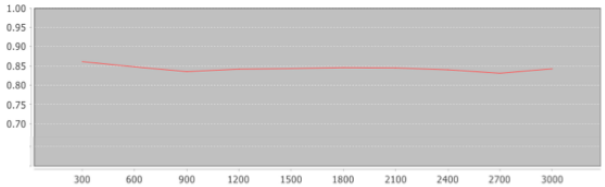

図 1: 提案手法で求めた木の大きさと LCST との比較

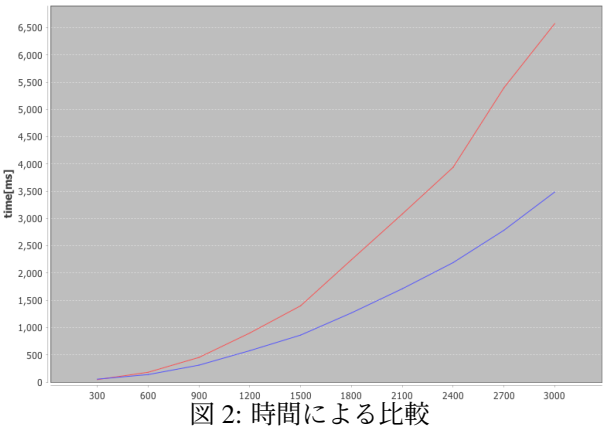

図 2: 時間による比較

縦軸はかかった時間を表している．また，赤線が Klein のアルゴリズム，青線が提案手法である．

本実験において，バンディットアルゴリズムは UCB1[2] を利用した．そして，本実験に使用した木構造データは，以下を繰り返すことによって生成している．

ノードを 1 つ生成する．このノードを v とし，v のラベルをランダムに決定する．

1. 生成されている木のサイズが 0 ならば，v の親と子供には何も指定せず，v がこの木の根となる．
2. 生成されている木のサイズが 1 以上ならば，その木のノード集合の中からにランダムに 1 つノード w を選択し，w の一番右の子供に v を追加する．

また，本実験に用いたプログラムは，Java version ”1.7.0_71” を用いて実装した．また実行した PC 環境については，プロセッサ 1.7GHz，Core i 7，メモリは 8GB，1600MHz，DDR3 である．

5 まとめ

本研究では，バンディットアルゴリズムを利用した Tai マッピングに基づく近似的最大共通部分木の計算の手法を提案し，実装，実験を行った．本提案手法は，Klein のアルゴリズムの半分程度の時間で最大共通部分木の 85%程度の大きさの共通部分木を求めることができることが示せた．

参考文献

[1] K.Tai. The tree-to-tree correction problem. *Journal of Association of Computing Machinery*, Vol. 26, pp. 422–433, 1979.
[2] P.Auer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, Vol. 47, No. 2-3, pp. 235–256, May 2002.
[3] P.Klein. Computing the edit-distance betwee unrooted ordered trees. In *In Proceedings of the 6th annual European Symposium on Algorithms*, pp. 91–102, 1998.