人工知能学会研究会資料
SIG-FPAI-B502-11

根付き木のアンカーアライメントの実装

On Computing Anchored Alignment Distance for Rooted Trees

石坂 悠眞[1*]　芳野 拓也[1] 平田 耕一[2]
Yuma Ishizaka[1]　Takuya Yoshino[1]　Kouichi Hirata[2]

[1] 九州工業大学情報工学府　[2] 九州工業大学情報工学研究院
[1] Graduate School of Computer Science and Systems Engineering
[2]Department of Artificial Intelligence
Kyushu Institute of Technology

Abstract: An anchored alignment is the problem, when two rooted labeled trees and an anchoring as a mapping are given as input, to output an anchored alignment tree if there exists and to return "no" otherwise. In this paper, for rooted unordered labeled trees with the same size of leaves, by providing the anchoring as the set of all of the leaves, we implement computing an anchored alignment distance as the minimum cost of anchored alignment trees and compare the previous distance as top-down, LCA-preserving and constrained distances. Furthermore, we improve the computation by incorporating the anchored alignment with the top-down mapping.

1　はじめに

木構造はさまざまなアルゴリズムで重要な役割を果たし, 構文木や進化系統樹, 生物の糖鎖構造の表現にも用いられるデータ構造である. これらのデータは根付き木 (以後, 単に木という) として扱い, 2 つの木が与えられたとき, それらの構造がどの程度似ているかという指標には, さまざまな類似度や非類似度 (距離) が提案されており, その 1 つにアライメント木の最小コストとなる**アライメント距離** (alignment distance)[2] がある. アライメント距離の計算は, 木のノードの個数を n とすると, 順序木の計算は $O(n^4)$ 時間であり, 無順序木の計算は NP 困難であることが知られている. そこで, 効率的に距離を求めるために, 様々な距離の変種が提案されている [6, 7, 8, 9].

アライメント距離の計算により木のノード間の対応付けであるアンカー (マッピング) を得ることができ, このアンカーから 2 つの木のアライメント木を構成することができる. アンカーに含まれるノードの組を含むアライメント木を**アンカーアライメント木** (anchored alignment tree) という. ここで, 2 つの木がアライメント木を持つための必要十分条件はアンカーが**アライメント可能マッピング** (alignable mapping) になることであり, アライメント可能マッピングは**劣制限マッピング** (less-constrained mapping)[4] と一致する [3].

*連絡先： 九州工業大学情報工学符
〒 820-8502 福岡県飯塚市川津 680-4
E-mail: y_ishizaka@dumbo.ai.kyutech.ac.jp

本研究では, 以下の**アンカーアライメント問題** (anchored alignment problem) を計算するアルゴリズムを実装する.

> アンカーアライメント問題
> 入力：木 T_1, T_2 および T_1, T_2 の間のアンカー M
> 出力：M を含む T_1 と T_2 のアンカーアライメント木. 存在しないときは "no" を返す.

この問題を解くアルゴリズムの計算時間は, T_1 のノードの個数が n, T_2 のノードの個数が m, T_1 と T_2 の最大の高さが h, アンカーの個数が a のとき, 順序木, 無順序木ともに $O(h^2a + n + m)$ 時間である [1].

そして, 実装を無順序木とみなした糖鎖データに適応する. ここで, 無順序木のアライメント距離の計算は NP 困難であるため, アンカーアライメント木のコストを**アンカーアライメント距離** (anchored alignment distance) と定式化する. そして, それと**制限付き距離** (constrained distance)[6, 7], *LCA* **保存距離** (LCA-preserving distance)[8], **トップダウン距離** (top-down distance)[9] とを比較し, その結果を考察する.

2　準備

閉路を持たない連結無向グラフを**木** (tree) という. さらに, 木の中に含まれる木を**部分木** (sub tree) という. 木を構成する頂点のことを**ノード** (node) という. 木 T に含まれるノードの集合を $V(T)$ と表す.

1つのノード v を**根**として選んだ木を**根付き木** (rooted tree) といい, v を $r(T)$ と表す.

根付き木の根から頂点 v への経路上の頂点を, v の**先祖** (ancestors) という. v の先祖を u とすると, このときの関係を $v < u$ と表し, u が v を先祖に持つならば, u は v の**子孫** descendant であるという. また, 特に v に隣接する先祖を, v の**親** (parent) といい, $pare(v)$ と表す. u が v の親の時, v は u の**子** (child) という. また, 子の数を**次数** (degree) といい $deg(u)$ と表す. さらに, 頂点 u と v の**最近共通先祖** (least common ancestor) を $u \sqcup v$ と表す.

木 T における頂点 $w \in T$ に対して, w の**先行順** (pre-order) における順番を $pre(w)$, **後行順** (post-otder) における順番を $post(w)$ と表す. このとき, $u \in T$ に対して, $pre(u) < pre(w)$ かつ $post(u) < post(w)$ のとき, u は w の**左にある**といい, w は u の**右にある**という.

各頂点がアルファベット Σ でラベル付けされている木を, **ラベル付き木** (labeles tree) といい, 頂点 v のラベルを $l(v)$ と表す. また, 各頂点の子に左右の順序を指定している根付き木を, **根付き順序木** (rooted ordered tree) といい, 順序を指定していない根付き木を, **根付き無順序木** (rooted unordered tree) という. 本稿では, 根付きラベル付き順序木を**順序木**といい, 根付きラベル付き無順序木を**無順序木**という.

空白記号 $\varepsilon \notin \Sigma$ に対して, $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ とする. このとき, ラベルの組に対する**コスト関数** (cost function) を $\gamma\colon (\Sigma_\varepsilon \times \Sigma_\varepsilon - \{(\varepsilon,\varepsilon)\}) \mapsto \mathbf{R}$ とする.

定義 1 (*Tai* **マッピング** [5]). T_1, T_2 を木とすると, $M \subseteq V(T_1) \times V(T_2)$ 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. $v_1 = v_2 \Leftrightarrow w_1 = w_2$.
2. v_1 が v_2 の先祖 $\Leftrightarrow w_1$ が w_2 の先祖.
3. v_1 が v_2 の左にある $\Leftrightarrow w_1$ が w_2 の左にある.

また, 条件1と2を満たすマッピングを**無順序木の** *Tai* **マッピング**という.

本稿では, Tai マップングを単に**マッピング**という. マッピングには以下のようにコストを割り振ることができる. ここで, I および J を, マッピング M の要素ではない T_1 の頂点の集合および M の要素ではない T_2 の頂点の集合とする.

$$\gamma(M) = \sum_{(v,w)\in M} \gamma(v \mapsto w) + \sum_{v\in I} \gamma(v \mapsto \varepsilon) + \sum_{w\in J} \gamma(\varepsilon \mapsto w).$$

定義 2 (変種). M を木 T_1 と T_2 の間のマッピングとする.

1. 任意の $(v_1,w_1),(v_2,w_2),(v_3,w_3) \in M$ が以下の条件を満たすとき, M を**劣制限マッピング**という.

$$v_1 \sqcup v_2 < v_1 \sqcup v_3 \Longrightarrow w_2 \sqcup w_3 = w_1 \sqcup w_3.$$

また, 木 T_1, T_2 の**アライメント距離** (alignment distance) $\tau_{less}(T_1,T_2)$ を以下のように定義する.

$$\tau_{less}(T_1,T_2) = \min\{\gamma(M) \mid M \text{ は劣制限マッピング}\}.$$

2. 任意の $(v_1,w_1),(v_2,w_2),(v_3,w_3) \in M$ が以下の条件を満たすとき, M を**制限マッピング**という.

$$v_3 < v_1 \sqcup v_2 \Leftrightarrow w_1 \sqcup w_2.$$

また, 木 T_1, T_2 の**制限付き距離** (constrained distance) $\tau_{const}(T_1,T_2)$ を以下のように定義する.

$$\tau_{const}(T_1,T_2) = \min\{\gamma(M) \mid M \text{ は制限マッピング}\}.$$

3. 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, M を *LCA* **保存マッピング**という.

$$(v_1 \sqcup v_2, w_1 \sqcup w_2) \in M.$$

また, 木 T_1, T_2 の *LCA* **保存距離** (LCA-preserving distance) $\tau_{LCA}(T_1,T_2)$ を以下のように定義する.

$$\tau_{LCA}(T_1,T_2) = \min\{\gamma(M) \mid M \text{ は LCA 保存マッピング}\}.$$

4. 任意の $(v,w) \in M \setminus \{r(T_1), r(T_2)\}$ が以下の条件を満たすとき, M を**トップダウンマッピング**という.

$$(pre(v), pre(w)) \in M.$$

また, 木 T_1, T_2 の**トップダウン距離** (top-down distance) $\tau_{top}(T_1,T_2)$ を以下のように定義する.

$$\tau_{top}(T_1,T_2) = \min\{\gamma(M) \mid M \text{ はトップダウンマッピング}\}.$$

3 アンカーアライメント問題を計算するアルゴリズム

本節では, アンカーアライメント問題を計算するアルゴリズムについて説明する.

定義 3 (**被覆集合**, **被覆列** [1]). T を r を根とする木, $T[v]$ を v を根とする部分木とし, U を T のノードの集合とする. また, $v \in T$ に対して, $UP_r(v)$ を v から r までのパス $(v = v_1, \ldots, v_n = r)$ とする. U に関する $v \in T$ の**被覆集合** (cover set) を $\{w \in T[v] \mid w \in U\}$ と定義し, $C_T(v, U)$ と表す. また, U に関する $v \in T$ の**被覆列** (cover sequence) を $C_i = C_T(v_i, U)$ を並べた列 $C_1, \ldots, C_n$ と定義し, $S_T(v, U)$ と表す.

補題 1. T_1 と T_2 を木とし, M を T_1 と T_2 の間のアライメント可能マッピングとする. $(v, w) \in M$ について, $S_{T_1}(v, M|_1)$ と $S_{T_2}(w, M|_2)$ は, 最後の要素がそれぞれ $M|_1$ と $M|_2$ になる比較可能増加列である.

M を T_1 と T_2 の間のマッピングとする. 任意の $(v_j, w_j) \in M$ について, j によって $v_j \in M|_1$ と $w_j \in M|_2$ を識別する. この識別下において, $M|_1 = M|_2$ とみなすことができる. 以下に, 比較可能増加列であるアラインド列とアラインドパスについて説明する.

定義 4 (**アライン列**, **アラインパス** [1]). $S_1 = A_1, \ldots, A_n$ と $S_2 = B_1, \ldots, B_m$ を $A_n = B_m$ となる比較可能増加列とする. アルゴリズム 1 の AlnSq によって求まる, $S_1' = A_1', \ldots, A_k'$ と $S_2' = B_1', \ldots, B_k'$ をそれぞれ S_1 と S_2 の**アライン列** (aligned sequence) と定義する.

procedure AlnSq(S_1, S_2)
  /* $S_1 = A_1, \ldots, A_n$, $S_2 = B_1, \ldots, B_m$ */
  $i \leftarrow 1$; $j \leftarrow 1$; $k \leftarrow 1$;
  while $i = n+1$ **and** $j = m+1$ **do**
    if $i = n+1$ **then**
      $A_k' \leftarrow \lambda$; $B_k' \leftarrow B_j$; j++;
    else if $j = m+1$ **then**
      $A_k' \leftarrow A_i$; $B_k' \leftarrow \lambda$; i++;
    else if $A_i = B_j$ **then**
      $A_k' \leftarrow A_i$; $B_k' \leftarrow B_j$; i++; j++;
    else if $A_i \subset B_j$ **then**
      $A_k' \leftarrow A_i$; $B_k' \leftarrow \lambda$; i++;
    else if $A_i \supset B_j$ **then**
      $A_k' \leftarrow \lambda$; $B_k' \leftarrow B_j$; j++;
    k++;
  return $S_1' = A_1', \ldots, A_k'$ and $S_2' = B_1', \ldots B_k'$;

アルゴリズム 1: AlnSq.

$S_1' = A_1', \ldots, A_k'$ と $S_2' = B_1', \ldots, B_k'$ を S_1 と S_2 のアライン列とする. $V = \{p_1, \ldots, p_k\}$, $E = \{(p_i, p_{i+1}) \mid 1 \leq i \leq k-1\}$ とし, p_1 を P の根, p_i のラベルを $(A_{k-i+1}', B_{k-i+1}')(1 \leq i \leq k)$ とする根付きラベル付きパス $P = (V, E)$ を S_1 と S_2 の**アラインパス** (aligned path) と定義する.

アンカーアライメント問題は以下の手順で計算することができる [1].

Step1. 入力木 T_1, T_2 および T_1, T_2 の間のアンカー M から, すべての $(v_j, w_j) \in M$ に対してアンカーに関する被覆列 $S_{T_1}(v_j, M|_1)$ と $S_{T_2}(w_j, M|_2)$ を計算する.

Step2. $S_{T_1}(v_j, M|_1)$ と $S_{T_2}(w_j, M|_2)$ からアラインパス $P_M(j)$ を計算する.

Step3. すべての $P_M(j)$ に対して, 同じラベルの要素を合わせることで, 木を構築する.

Step4. $UP_{r_1}(v_j)$ と $UP_{r_2}(w_j)$ に含まれないノードを Step3 で構築した木に追加することで, 入力木 T_1, T_2 のアンカーアライメント木を求める.

定義 5 (**アンカーアライメント木のコスト**). T をアンカーアライメント木とし, ラベルに関するコスト関数を $\gamma\colon (\Sigma_\varepsilon \times \Sigma_\varepsilon - \{(\varepsilon, \varepsilon)\}) \to \mathbf{R}^+$ とする. このとき, **アンカーアライメント木のコスト** $\gamma(T)$ を以下のように定義する.

$$\gamma(T) = \sum_{v \in T} \gamma(v).$$

定義 6 (**アンカーアライメント距離**). 木 T_1 と T_2 の**アンカーアライメント距離** (anchored alignment distance) を $\tau_{anchor}(T_1, T_2)$ を以下のように定義する.

$$\tau_{anchor}(T_1, T_2) = \{\gamma(T) \mid T \text{ は } T_1 \text{と } T_2 \text{のアンカーアライメント木 }\}.$$

4 実験

本節では, 根付き木のアンカーアライメントの実装に対する実験結果について述べる. ここでは, データとして糖鎖 (glycan) データを用いる. 実験環境は, OS が Microsoft Windows 7, CPU が Intel(R) Core(TM) i7 3.5GHz, メモリが 32GB である.

実験は, 糖鎖データを木の葉の数で分割し, 同じ数の葉を持つ木を組にして, アンカーアライメント距離 τ_{anchor}, 制限付き距離 τ_{const}, LCA 保存距離 τ_{LCA}, トップダウン距離 τ_{top} を計算し, その距離を比較する. ここで, 葉のみを全てマッピングしたものを**葉マッピング**とする. アンカーアライメント距離の計算は, 入力木の組のマッピングを, 葉マッピングの全パタンと葉マッピングの全パタンにトップダウンマッピングを合わせたものとした 2 通りで行った. 図 1 は, 葉マッピングと葉マッピングにトップダウンマッピングを合わせたマッピングの例である.

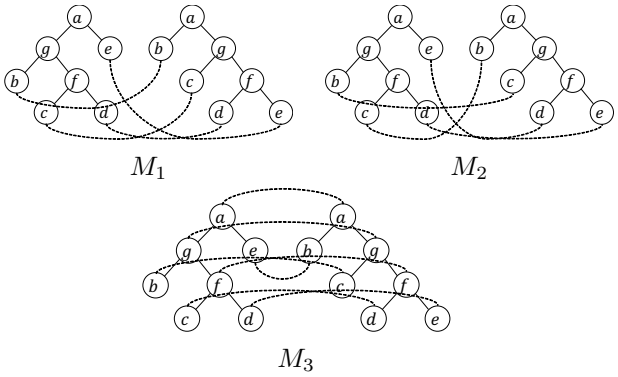

図 1: M_1, M_2 は葉マッピング, M_3 は葉マッピングにトップダウンマッピングを組み合わせたマッピング.

表 1 は, アンカーアライメント距離の入力のマッピングを葉マッピングの全パタンとした場合の τ_{anchor} と τ_{const}, τ_{LCA}, τ_{top} の大小関係を比較した結果である. 表 2 は, アンカーアライメント距離の入力のマッピングを葉マッピングの全パタンにトップダウンマッピングを合わせたものとした場合の τ_{anchor} と τ_{const}, τ_{LCA}, τ_{top} の大小関係を比較した結果である. なお, 表 1, 2 において, AA はアンカーアライメント距離, Con は制限付き距離, LCA は LCA 保存距離, Top はトップダウン距離とする.

表 1 と表 2 より, 全ての木の組数に対する AA が求まる組数の合計は, アンカーアライメント距離の入力のマッピングを葉マッピングの全パタンとした場合で 30% 多いことが分かる. また, τ_{anchor} と τ_{const}, τ_{LCA}, τ_{top} の大小関係の比較では, アンカーアライメント距離の入力のマッピングを葉マッピングの全パタンにトップダウンマッピングを合わせたものとした場合が, AA≤Top となる組数の合計で 12%, AA≤LCA となる組数の合計で 22%, AA≤Con となる組数の合計で 23% 多いことが分かる.

図 2 から図 4 はアンカーアライメント距離の入力マッピングを, 葉マッピングとした場合の τ_{anchor} と τ_{const}, τ_{LCA}, τ_{top} の分布を表した図である. 図 5 から図 7 はアンカーアライメント距離の入力マッピングを, 葉マッピングの全パタンにトップダウンマッピングを合わせたものとした場合の τ_{anchor} と τ_{const}, τ_{LCA}, τ_{top} の分布を表した図である. 図 2 から図 7 において, 斜線は距離同士が等しいことを表している. また, 点の直径と色は, 木の組数の割合を表しており, 同じ点に多くの組が存在した場合は, 直径は大きくなり, 色は濃くなる.

図 2 から図 4 より, アンカーアライメント距離の入力マッピングを, 葉マッピングとした場合の τ_{anchor} と τ_{const}, τ_{LCA} の分布では, 斜線の上部の色が濃くなっており, τ_{anchor} と τ_{top} の分布では, 斜線周辺の色が濃くなっていることが分かる. 一方, 図 5 から図 7 より, アンカーアライメント距離の入力マッピングを, 葉マッピングの全パタンにトップダウンマッピングを合わせたものとした場合では, 斜線の下部の色が濃くなっていることが分かる. これは, 葉マッピングにトップダウンマッピングを合わせたことによる効果だと考えられる.

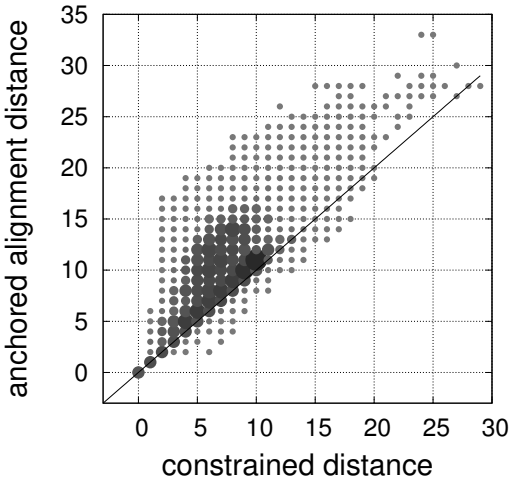

図 2: アンカーアライメント距離の入力マッピングを, 葉マッピングとした場合の τ_{anchor} と τ_{const} の分布.

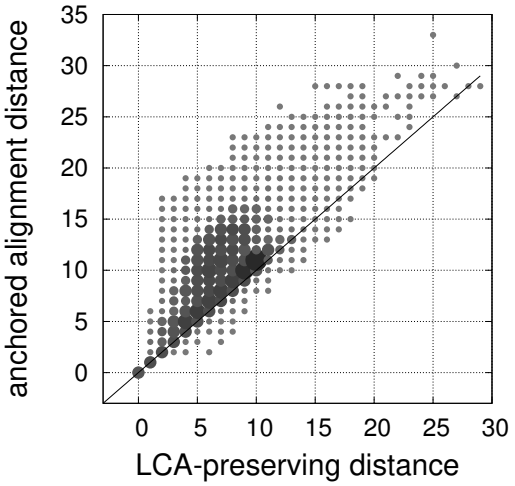

図 3: アンカーアライメント距離の入力マッピングを, 葉マッピングとした場合の τ_{anchor} と τ_{LCA} の分布.

表 1: アンカーアライメント距離の入力マッピングを, 葉マッピングとした場合の他の距離との比較データ.

葉数	木の組数	AA の組数	AA>Top	AA=Top	AA<Top	AA>LCA	AA=LCA	AA<LCA	AA>Con	AA=Con	AA<Con
3	233586	233586	113095	25702	94789	189810	38004	5772	190657	37278	5651
4	148240	94404	47091	8699	38614	77251	14927	2226	77987	14273	2144
5	29890	21325	10769	2172	8385	18331	2759	235	18576	2516	233
6	1225	785	380	92	313	649	90	46	676	65	44
7	120	40	17	2	21	33	7	0	34	6	0
8	45	13	13	0	0	13	0	0	13	0	0
9	6	3	1	0	2	2	1	0	2	1	0
10	6	2	1	1	0	1	1	0	1	1	0
合計	413118	350158	171367	36668	142124	286090	55789	8279	287946	54140	8072

表 2: アンカーアライメント距離の入力マッピングを, 葉マッピングの全パタンにトップダウンマッピングを合わせたものとした場合の他の距離との比較データ.

葉数	木の組数	AA の組数	AA>Top	AA=Top	AA<Top	AA>LCA	AA=LCA	AA<LCA	AA>Con	AA=Con	AA<Con
3	233586	132318	5369	31949	95000	45245	27043	60030	45395	27105	59818
4	148240	78285	836	11775	65674	19897	12527	45861	20088	12570	45627
5	29890	15717	31	2539	13147	2764	2859	10094	2832	2851	10034
6	1225	595	3	67	525	116	81	398	125	98	372
7	120	69	0	16	53	14	19	36	15	20	34
8	45	23	0	7	16	1	7	15	1	7	15
9	6	0	0	0	0	0	0	0	0	0	0
10	6	2	0	2	0	0	2	0	0	2	0
合計	413118	227009	6239	46355	174415	68037	42538	116434	68456	42653	115900

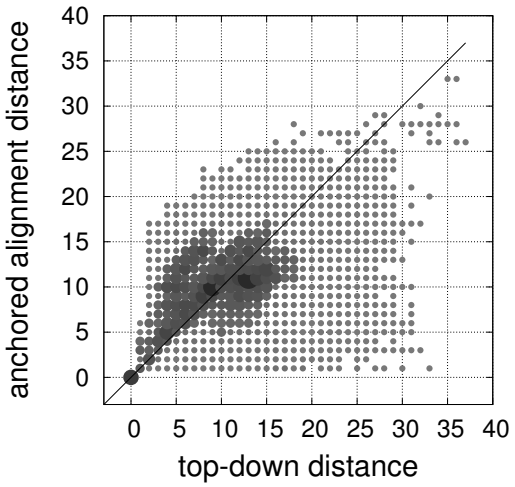

図 4: アンカーアライメント距離の入力マッピングを, 葉マッピングとした場合の τ_{anchor} と τ_{top} の分布

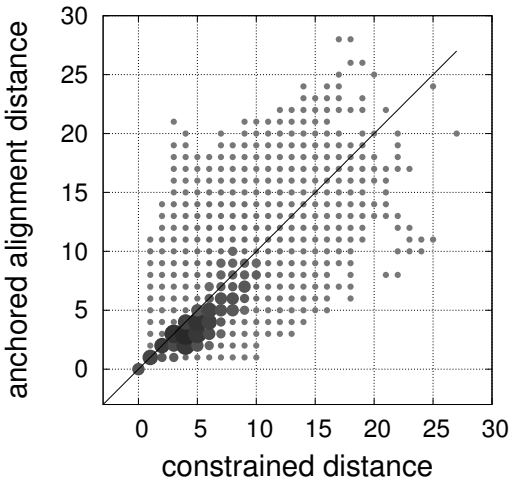

図 5: アンカーアライメント距離の入力マッピングを, 葉マッピングの全パタンにトップダウンマッピングを合わせたものとした場合の τ_{anchor} と τ_{const} の分布.

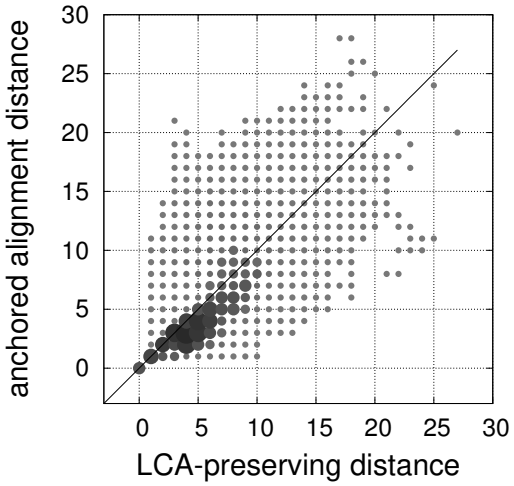

図 6: アンカーアライメント距離の入力マッピングを, 葉マッピングの全パタンにトップダウンマッピングを合わせたものとした場合の τ_{anchor} と τ_{LCA} の分布.

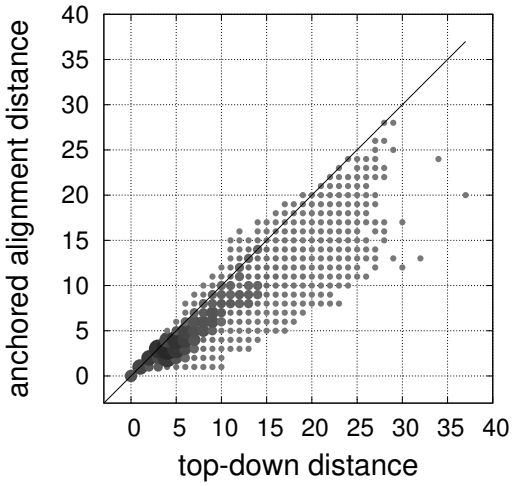

図 7: アンカーアライメント距離の入力マッピングを, 葉マッピングの全パタンにトップダウンマッピングを合わせたものとした場合の τ_{anchor} と τ_{top} の分布.

5 まとめ

本稿では, 根付き木のアンカーアライメントの実装を行い, 糖鎖データを用いて, アンカーアライメント距離と制限付き距離, LCA 保存距離, トップダウン距離との比較を行った. 比較した結果から, アンカーアライメントの入力マッピングを葉マッピングとした場合, アンカーアライメント距離が求まる組数が多いことが分かった. また, 距離の比較では, 入力マッピングを葉マッピングにトップダウンマッピングを組み合わせたマッピングとした場合, 制限付き距離, LCA 保存距離, トップダウン距離以下となる組数が多いことが分かった.

今回, 同じ葉の数の木間のアンカーアライメント距離を計算した. 今後の課題として, 葉の数が異なる木間のアンカーアライメント距離の計算の実装が挙げられる.

参考文献

[1] Y. Ishizaka, T. Yoshino, K. Hirata: *Anchor Alignment Problem for Rooted Labeled Trees*, New Frontiers in Artificial Intelligence. LNAI. **9067**, 296–309, 2015.

[2] T. Jiang, L. Wang, K. Zhang: *Alignment of trees – an alternative to tree edit*, Theoret. Comput. Sci. **143**, 137–148, 1995.

[3] T. Kuboyama: *Matching and learning in trees*, Ph.D thesis, University of Tokyo, 2007.

[4] C. L. Lu, Z.-Y. Su, C. Y. Yang: *A new measure of edit distance between labeled trees*, Proc. COCOON'01, LNCS **2108**, 338–348, 2001.

[5] K.-C. Tai: *The tree-to-tree correction problem*, J. ACM **26**, 422–433, 1979.

[6] K. Zhang: *Algorithms for the constrained editing distance between ordered labeled trees and related problems*, Patter Recog **28**, 463–474, 1995.

[7] K. Zhang: *A constrained edit distance between unordered labeled trees*, Algorithmica **15**, 205–222, 1996.

[8] K. Zhang, J. Wang, D. Shasha: *On the editing distance between undirected acyclic graphs*, Int. J. Found. Comput. Sci. **7**, 43–58, 1995.

[9] S. M. Selkow: *The tree-to-tree editing problem*, Inform Process **6**, 184–186, 1977.