Caterpillar Inclusion: Inclusion Problem for Rooted Labeled Caterpillars

Tomoya Miyazaki, Manami Hagihara and Kouich Hirata
Kyushu Institute of Technology, Kawazu 680-4, Iizuka 820-8502, Japan

Keywords: Caterpillar Inclusion, Tree Inclusion, Rooted Labeled Caterpillar, Rooted Labeled Unordered Tree.

Abstract: In this paper, we investigate an inclusion problem for *rooted labeled caterpillars* (*resp.*, *caterpillars*, for short), which we call a *caterpillar inclusion*. The caterpillar inclusion is to determine whether or not a *text caterpillar* T achieves to a *pattern caterpillar* P by deleting vertices in T. Then, we design the algorithm of the caterpillar inclusion for P and T in $O((h+H)\sigma)$ time, where h is the height of P, H is the height of T and σ is the number of labels occurring in P and T. Also we give experimental results for the algorithm by using real data for caterpillars.

1 INTRODUCTION

The pattern matching for tree-structured data such as HTML and XML documents for web mining or DNA and glycan data for bioinformatics is one of the fundamental tasks for information retrieval or query processing. As such pattern matching for *rooted labeled unordered trees*, an *unordered tree inclusion* (an *inclusion*, for short) is the problem of determining whether or not an unordered tree P called a *pattern tree* is included in an unordered tree T called a *text tree*, that is, T achieves to P by deleting vertices in T. However, the inclusion is known to be intractable, that is, NP-complete (Kilpeläinen and Mannila, 1995; Matoušek and Thomas, 1992).

In order to overcome such intractability, several researches have developed the tractable variations of the inclusion such as a *top-down* inclusion (Shamir and Tsur, 1999), a *bottom-up* inclusion (Valiente, 2002), an *LCA-preserving* inclusion (Valiente, 2005)[1] and an *isolated-subtree inclusion* (Hokazono et al., 2012). The first three variations are formulated by restricting the scope of the deletion of vertices to just leaves, just roots and just either leaves or vertices with one child, respectively. Also the top-down (*resp.*, bottom-up) inclusion coincides with the top-down (*resp.*, bottom-up) unordered subtree isomorphism (*cf.*, (Valiente, 2002)). On the other hand, the several algorithms to compute unordered tree inclusion have been designed as the exact exponential algorithms (Akutsu et al., 2021; Kilpeläinen and Mannila, 1995).

Note that the proof of NP-completeness for the inclusion implies the structural restriction of the tractability for the inclusion that the height of a text tree is at most 2 or the degree of a text tree is bounded by some constant (Kilpeläinen and Mannila, 1995; Matoušek and Thomas, 1992). In this paper, we give another structural restriction providing the limitation of the tractability for the inclusion as a *rooted labeled caterpillar* (a *caterpillar*, for short) (*cf.*, (Gallian, 2007)). The caterpillar is an unordered tree transformed to a rooted path after removing all the leaves in it.

The caterpillar provides the structural restriction of the tractability of computing the *edit distance* for unordered trees (Muraka et al., 2018). It is known that the problem of computing the edit distance between unordered trees is MAX SNP-hard (Zhang and Jiang, 1994). This statement also holds even if two trees are binary, the maximum height is at most 3 or the cost function is the unit cost function (Akutsu et al., 2013; Hirata et al., 2011). On the other hand, we can compute the edit distance between caterpillars in $O(h^2\lambda^3)$ time in the general cost function and $O(h^2\lambda)$ time under the unit cost function, where h is the maximum height of the two caterpillars and λ is the maximum number of leaves in the two caterpillars (Muraka et al., 2018)[2].

[1]While Valiente (Valiente, 2005) has called it a *constrained* inclusion, the definition of (Valiente, 2005) is corresponding to an LCA-preserving distance or a degree-2 distance (Zhang et al., 1996). Hence, we call it an *LCA-preserving* inclusion.

[2]This time complexity is different from the result in (Muraka et al., 2018), because it contains some errors. See

Miyazaki, T., Hagihara, M. and Hirata, K.
Caterpillar Inclusion: Inclusion Problem for Rooted Labeled Caterpillars.
DOI: 10.5220/0010826300003122
In *Proceedings of the 11th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2022)*, pages 280-287
ISBN: 978-989-758-549-4; ISSN: 2184-4313

Hence, in this paper, we investigate a *caterpillar inclusion* of determining whether or not a pattern caterpillar P is included in a text caterpillar T. Then, we design the algorithm CATINC of determining whether or not P is included in T in $O((h+H)\sigma)$ time, where h is the height of P, H is the height of T and σ is the number of labels occurring in P and T. Also, we give experimental results for the algorithm CATINC by using real caterpillar data.

2 PRELIMINARIES

A *tree* is a connected graph without cycles. For a tree $T = (V,E)$, we denote V and E by $V(T)$ and $E(T)$. We sometimes denote $v \in V(T)$ by $v \in T$. A *rooted tree* is a tree with one vertex r chosen as its *root*, which we denote by $r(T)$.

For each vertex v in a rooted tree with the root r, let $UP_r(v)$ be the unique path from v to r. The *parent* of $v(\neq r)$ is its adjacent vertex on $UP_r(v)$ and the *ancestors* of $v(\neq r)$ are the vertices on $UP_r(v) - \{v\}$. We denote $u < v$ if v is an ancestor of u, and we denote $u \leq v$ if either $u < v$ or $u = v$. The parent and the ancestors of the root r are undefined. Also we say that w is the *least common ancestor* of u and v, denoted by $u \sqcup v$, if $u \leq w$, $v \leq w$ and there exists no w' such that $u \leq w'$, $v \leq w'$ and $w' \leq w$. We say that u is a *child* of v if v is the parent of u, and u is a *descendant* of v if v is an ancestor of u. We denote the set of all children of v by $ch(v)$. Two vertices with the same parent are called *siblings*. A *leaf* is a vertex having no children and we denote the set of all the leaves in T by $lv(T)$. We call a vertex that is neither the root nor a leaf an *internal vertex*. The *height* $h(v)$ of a vertex v is defined as $|UP_r(v)| - 1$ and the *height* $h(T)$ of T is the maximum height for every vertex $v \in T$.

We say that a rooted tree is *ordered* if a left-to-right order among siblings is given; *Unordered* otherwise. For a fixed finite alphabet Σ, we say that a tree is *labeled* over Σ if each vertex is assigned a symbol from Σ. We denote the label of a vertex v by $l(v)$, and sometimes identify v with $l(v)$. In this paper, we call a rooted labeled unordered tree over Σ a *tree*, simply.

Definition 1 (Isomorphic trees)**.** Let T_1 and T_2 be trees. Then, we say that T_1 and T_2 are *isomorphic*, denoted by $T_1 \cong T_2$, if there exists a set $F \subseteq V(T_1) \times V(T_2)$ satisfying the following conditions[3].

1. $\forall v \in V(T_1) \exists w \in V(T_2)$
   $\Big(((v,w) \in F) \wedge (l(v) = l(w))\Big)$.
   (inclusion condition)
2. $\forall w \in V(T_2) \exists v \in V(T_1)$
   $\Big(((v,w) \in F) \wedge (l(v) = l(w))\Big)$.
   (exclusion condition)
3. $\forall (v_1,w_1),(v_2,w_2) \in F$
   $\Big((v_1 = v_2) \iff (w_1 = w_2)\Big)$.
   (one-to-one condition)
4. $\forall (v_1,w_1),(v_2,w_2) \in F$
   $\Big((v_1 \leq v_2) \iff (w_1 \leq w_2)\Big)$.
   (ancestor condition)

As the restricted form of trees, we introduce a *rooted labeled caterpillar* (*caterpillar*, for short) as follows.

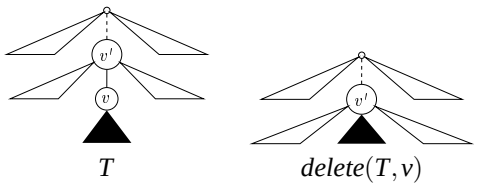
Definition 2 (Caterpillar)**.** We say that a tree is a *caterpillar* (*cf.* (Gallian, 2007)) if it is transformed to a rooted path after removing all the leaves in it. For a caterpillar C, we call the remained rooted path a *backbone* of C and denote it by $bb(C)$.

It is obvious that $r(C) = r(bb(C))$ and $V(C) = V(bb(C)) \cup lv(C)$ for a caterpillar C, that is, every vertex in a caterpillar is either a leaf or an element of the backbone.

3 TREE INCLUSION

In this section, we formulate a tree inclusion. Throughout of this paper, we just deal with an *unordered tree inclusion*, that is, the tree inclusion between unordered trees. Hence, we call an *unordered tree inclusion* an *inclusion* simply.

For a tree T and a vertex $v \in T$, the *deletion* of v in T is to delete a non-root vertex v in T with a parent v', making the children of v become the children of v' that are inserted in the place of v as a subset of the children of v'. We denote the result of the deletion of v in T by $delete(T,v)$. See the following figure.



Definition 3 (Inclusion)**.** Let P and T be trees. We sometimes call P a *pattern* tree and T a *text* tree. Then, we say that P is an *inclusion* of T, denoted by $P \sqsubseteq T$, if either $P \cong T$ or there exists a sequence of vertices $v_1,\ldots,v_k$ in T such that $T_0 \cong T$, $T_k \cong P$ and $T_{i+1} \cong delete(T_i, v_{i+1})$ $(0 \leq i \leq k-1)$.

(Ukita et al., 2021) in more detail.

[3]In this definition, we use the notion like a Tai mapping as Definition 4.

We can characterize the tree inclusion by using the following *inclusion mapping* (Hokazono et al., 2012), which is a variant of a Tai mapping in the tree edit distance (Tai, 1979).

Definition 4 (Inclusion mapping)**.** Let P and T be trees. Then, we say that the triple (I,P,T) is an (*unordered*) *inclusion mapping* from P to T if $I \subseteq V(P) \times V(T)$ satisfies the conditions 1, 3 and 4 in Definition 1, that is:

1. $\forall v \in V(P) \exists w \in V(T)$ $\Big(((v,w) \in I) \wedge (l(v) = l(w))\Big)$. (inclusion condition)
2. $\forall (v_1,w_1),(v_2,w_2) \in I$ $\Big((v_1 = v_2) \iff (w_1 = w_2)\Big)$. (one-to-one condition)
3. $\forall (v_1,w_1),(v_2,w_2) \in I$ $\Big((v_1 \leq v_2) \iff (w_1 \leq w_2)\Big)$. (ancestor condition)

We will use I instead of (I,P,T) when there is no confusion.

It is obvious that $P \sqsubseteq T$ if and only if there exists an inclusion mapping from P to T, and $P \cong T$ if and only if $P \sqsubseteq T$ and $T \sqsubseteq P$.

Theorem 1. (Kilpeläinen and Mannila, 1995) *For trees P and T, the problem of determining whether or not $P \sqsubseteq T$ is NP-complete. This statement also holds even if the maximum height of T is at most* 3.

4 CATERPILLAR INCLUSION

In this section, we investigate a *caterpillar inclusion* as the inclusion such that both a pattern tree P and a text tree T are caterpillars.

Let P and T be caterpillars such that $r(P) = v_1$ and $r(T) = w_1$. Then, we denote $bb(P) = [v_1,\ldots,v_n]$ for $(v_i,v_{i+1}) \in E(P)$ and $bb(T) = [w_1,\ldots,w_m]$ for $(w_j,w_{j+1}) \in E(T)$. Also L_i (*resp.*, M_j) denotes the set of leaves that are children of v_i (*resp.*, w_j) for $1 \leq i \leq n$ (*resp.*, $1 \leq j \leq m$).

In this section, we use a *multiset* of labels in order to compare two sets of vertices. A *multiset* on Σ is a mapping $S : \Sigma \to \mathbf{N}$. For a multiset S on Σ, we say that $a \in \Sigma$ is an *element* of S if $S(a) > 0$. An *empty* multiset is a multiset S such that $S(a) = 0$ for every $a \in \Sigma$ and denote it by $\emptyset$ (like as a standard set). Let S_1 and S_2 be multisets on Σ. Then, we define the *intersection* $S_1 \sqcap S_2$ and the *difference* $S_1 \setminus S_2$ as multisets satisfying that $(S_1 \sqcap S_2)(a) = \min\{S_1(a), S_2(a)\}$ and $(S_1 \setminus S_2)(a) = \max\{S_1(a) - S_2(a), 0\}$ for every $a \in \Sigma$. Note that $S_1 \setminus S_2 = S_1 \setminus S_1 \sqcap S_2$. For a set V of vertices, we denote the multiset of labels occurring in V by $\widetilde{V}$.

Then, we design the algorithm CATINC in Algorithm 1.

```
    procedure CATINC(P,T)
        / P : caterpillar, bb(P) = [v_1,...,v_n] /
        / T : caterpillar, bb(T) = [w_1,...,w_m] /
1       if n > m then return "No"; halt;
2       else
3           j ← 1;
4           for i = 1 to n do
5               while l(v_i) ≠ l(w_j) do
6                   if j < m then j++;
7                   else return "No"; halt;
                / l(v_i) = l(w_j) /
8               N̂_i ← L̂_i \ M̂_j;
9               while ((i < n and N̂_i ≠ ∅) or
                (i = n and N̂_n \ {w_{j+1}}^~ ≠ ∅)) do
10                  if j < m then
11                      j++; N̂_i ← N̂_i \ M̂_j;
12                  else return "No"; halt;
13              j++;
14          return "Yes";
```

Algorithm 1: CATINC.

Example 1. Consider a pattern caterpillar P and a text caterpillar T in Figure 1, where $bb(P) = [v_1,v_2,v_3]$ and $bb(T) = [w_1,\ldots,w_6]$. Also we denote a multiset as a string, that is, we denote a multiset $\{a,a,b,c,c\}$ as a string a^2bc^2, for example. Then, we run the algorithm CATINC(P,T).

First, the algorithm CATINC(P,T) finds w_j such that $l(v_1) = l(w_j)$. Then, it sets $j = 1$ and computes $\widetilde{L_1} \setminus \widetilde{M_1} = bc \setminus b^2 = c = \widetilde{N_1}$. Since $\widetilde{N_1} \neq \emptyset$, after incrementing j as 2 it computes $\widetilde{N_1} \setminus \widetilde{M_2} = c \setminus c^2 = \emptyset$, which is set to $\widetilde{N_1}$. Since $\widetilde{N_1} = \emptyset$, the algorithm CATINC$(P,T)$ increments j as 3 and i as 2.

Secondly, the algorithm CATINC(P,T) finds w_j such that $l(v_2) = l(w_j)$ for $j \geq 3$. Then, it sets $j = 3$ and computes $\widetilde{L_2} \setminus \widetilde{M_3} = b^2 \setminus b^2c = \emptyset = \widetilde{N_2}$. Since $\widetilde{N_2} = \emptyset$, the algorithm CATINC$(P,T)$ increments j as 4 and i as 3.

Finally, the algorithm CATINC(P,T) finds w_j such that $l(v_3) = l(w_j)$ for $j \geq 4$. Then, it sets $j = 4$ and computes $\widetilde{L_3} \setminus \widetilde{M_4} = c^3 \setminus b^2c = c^2 = \widetilde{N_3}$. Since $\widetilde{N_3} \neq \emptyset$, after incrementing j as 5 it computes $\widetilde{N_3} \setminus \widetilde{M_5} = c^2 \setminus c = c$, which is set to $\widetilde{N_3}$. Since $\widetilde{N_3} \neq \emptyset$, after incrementing j as 6 it computes $\widetilde{N_3} \setminus \widetilde{M_6} = c \setminus bc =$

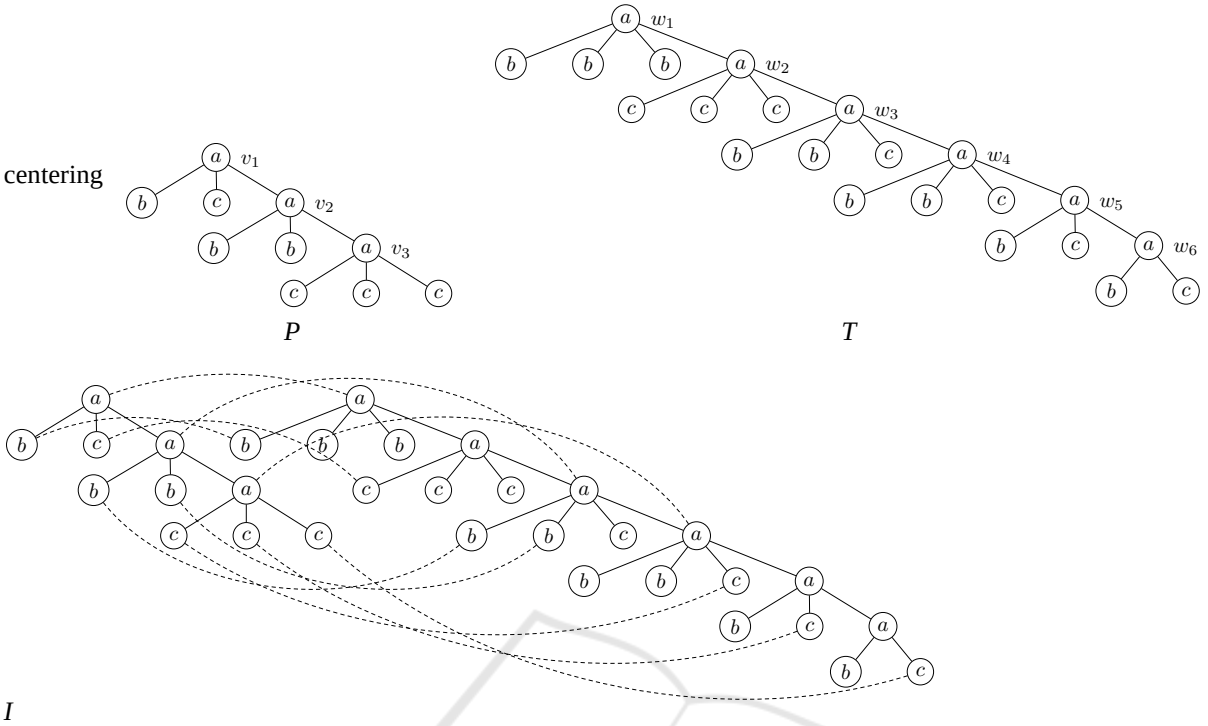


Figure 1: The pattern caterpillar P, the text caterpillar T and the inclusion mapping I from P to T in Example 1.

$\emptyset$, which is set to $\widetilde{N_3}$. Since $\widetilde{N_3} = \emptyset$, the algorithm CATINC$(P,T)$ returns "Yes".

We can obtain the inclusion mapping I from P to T consisting of the pairs $(v_1,w_1),(v_2,w_3),(v_3,w_4)$ for backbones such that $l(v_i) = l(w_j)$ at line 5, the correspondences between $\widetilde{L_i}$ and $\widetilde{M_j}$ in line 8 such that $\widetilde{L_i} \sqcap \widetilde{M_j}$ and the correspondences between $\widetilde{N_i}$ and $\widetilde{M_j}$ in line 10 such that $\widetilde{N_i} \sqcap \widetilde{M_j}$. Figure 1 illustrates the inclusion mapping I as the dashed line obtained by applying the algorithm CATINC(P,T).

Then, we can obtain the following main theorem of this paper.

Theorem 2. *Let P and T be caterpillars, where $h = h(P)$, $H = h(T)$ and $\sigma = |\Sigma|$. Then, we can determine whether or not $P \sqsubseteq T$ in $O((h+H)\sigma)$ time.*

Proof. First, we show the correctness of the algorithm CATINC. Let i be a current index in the for-loop of the algorithm CATINC. Also suppose that I is a mapping from P to T implicitly obtained by CATINC returned to "Yes."

Then, CATINC first finds j such that $l(v_i) = l(w_j)$ for $v_i \in bb(P)$ and $w_j \in bb(T)$, which implies that $(v_i,w_j) \in I$. Next, for i and j, CATINC corresponds the leaves in L_i to the leaves in M_j as possible, where the corresponding pair $(v,w) \in L_i \times M_j$ are added to I, and then store the remained (non-mapped) labels of leaves in $\widetilde{L_i}$ as $\widetilde{N_i}$, which is realized as the line 8. Then, until $\widetilde{N_i}$ is empty, CATINC increments j and finds the corresponding leaves in the next set M_j of leaves in T to N_i, where the found pair $(v,w) \in N_i \times M_j$ is added to I. Finally, when $i = n$, the condition in while-loop is changed that $\widetilde{N_n \cup \{w_{j+1}\}}$ is empty. This means that, when $\widetilde{N_n} \neq \emptyset$ and $\widetilde{N_n \setminus \{w_{j+1}\}} = \emptyset$, at most one vertex $w_{j+1} \in bb(T)$ is corresponding to a leaf $v \in L_n$ and (v,w_{j+1}) is implicitly added to I. In this case, the algorithm CATINC finishes and there exists no pair $(v,w) \in I$ such that w is a descendant of w_{j+1}.

For the obtained mapping I by the algorithm CATINC, it is obvious that I satisfies the inclusion condition and the one-to-one condition. Also, for pairs $(v_i,w_j),(v_{i+1},w_k) \in I$ such that $v_i,v_{i+1} \in bb(P)$ and $w_j,w_k \in bb(T)$ $(1 \leq i < n-1, 1 \leq j < k \leq m)$, every leaf in L_i is corresponding to a leaf in $M_j \cup \cdots \cup M_{k-1}$. In other words, there exist no pairs $(v,w),(v',w') \in I$ such that $v \in L_i$, $w \in M_j$, $v' \in L_{i'}$, $w' \in M_{j'}$, $i < i'$ and $j' < j$. Furthermore, for $i = n$, let $I_n = \{(v,w) \in I \mid v \in L_n\}$. Note that at most one vertex $w_{j+1} \in bb(T)$ is corresponding to a leaf in L_n. If such a w_{j+1} exists, then there exists no pair $(v,w) \in I_n$ such that w is a descendant of w_{j+1}. This implies that neither w is an ancestor of w' nor w' is an ancestor of w for every w and w' such that $w \neq w'$ and $(v,w),(v',w') \in I_n$. Then, I satisfies the ancestor condition. Hence, I is an inclusion mapping from P to T.

Next, we show the time complexity of the algo-

rithm CATINC. By using the hash table of Σ, we can process $\widetilde{N}_i \leftarrow \widetilde{L}_i \setminus \widetilde{M_j}$ and $\widetilde{N}_i \leftarrow \widetilde{N}_i \setminus \widetilde{M_j}$ in $O(\sigma)$ time. Also we can determine whether or not $\widetilde{N}_i \neq \emptyset$ in $O(\sigma)$ time. Furthermore, j changes from 1 to m in the for-loop while i changes from 1 to n in CATINC. Since $n = h$ and $m = H$, the algorithm CATINC runs in $O((h+H)\sigma)$ time. □

Theorem 2 also claims that the structural restriction of caterpillars provides the limitation of tractability of the tree inclusion problem. We say that a tree is a *generalized caterpillar* if it is transformed to a caterpillar after removing all the leaves in it. Then, Theorem 1 implies the following theorem.

Theorem 3. (Kilpeläinen and Mannila, 1995) *Let P be a caterpillar and T a generalized caterpillar. Then, the problem of determining whether or not $P \sqsubseteq T$ is NP-complete. This statement also holds even if the maximum height of T is at most* 3.

Theorem 3 is similar that the structural restriction of caterpillars provides the limitation of tractability of computing the edit distance. Whereas the problem of computing the edit distance between caterpillars is tractable as stated in Section 1, the problem of computing the edit distance between generalized caterpillars is MAX SNP-hard. This statement also holds even if the maximum height is at most 3 or the cost function is the unit cost function (Muraka et al., 2018).

5 EXPERIMENTAL RESULTS

In this section, we give the experimental results of computing CATINC. Here, the computer environment is that OS is Ubuntu 18.04.4, CPU is Intel Xeon E5-1650 v3(3.50GHz) and RAM is 3.8GB.

We deal with caterpillars for N-glycans and all-glycans from KEGG[4], CSLOGS[5], the largest 5,154 caterpillars (0.1%) in dblp[6] (refer to $\text{dblp}_{0.1\%}$), SwissProt and non-isomorphic caterpillars in TPC-H (refer to TPC-$\text{H}_\circ$) from UW XML Repository[7]. Also we deal with caterpillars obtained by deleting the root in Auction (refer to Auction$^-$) and non-isomorphic caterpillars obtained by deleting the root in Nasa (refer to NASA$^-_\circ$), Protein (refer to Protein$^-_\circ$) and University (refer to University$^-_\circ$) from UW XML Repository. Table 1 illustrates the information of such caterpillars. Here, #, n, d, h, λ and β are the number of caterpillars, the average number of vertices, the average degree, the average height, the average number of leaves and the average number of labels.

Table 1: The information of caterpillars.

data	#	n	d	h	λ	β
N-glycans	514	6.40	1.84	4.22	2.18	4.50
all-glycans	7,984	4.74	1.49	3.02	1.72	2.84
CSLOGS	41,592	5.84	3.05	2.20	3.64	5.18
$\text{dblp}_{0.1\%}$	5,154	41.74	40.73	1.01	40.73	10.61
SwissProt	6,804	35.10	24.96	2.00	33.10	16.79
TPC-$\text{H}_\circ$	8	8.63	7.63	1.00	7.63	8.63
Auction$^-$	259	4.29	3.00	0.71	3.57	4.29
Nasa$^-_\circ$	33	7.27	5.15	1.64	5.64	3.18
Protein$^-_\circ$	5,150	4,97	3.63	1.16	3.81	4.57
University$^-_\circ$	26	1.35	0.35	0.19	1.15	1.35

We compare all the pairs (P,T) in the caterpillars in Table 1. The number of pairs is $\# \times (\# - 1)$, and Table 2 summarizes such number as #pairs.

Table 2: The number (#pairs) of all the pairs in caterpillars in Table 1.

data	#pairs	data	#pairs
N-glycans	263,682	TPC-$\text{H}_\circ$	56
all-glycans	63,736,272	Auction$^-$	66,822
CSLOGS	1,729,852,872	Nasa$^-_\circ$	1,056
$\text{dblp}_{0.1\%}$	26,558,562	Protein$^-_\circ$	26,517,350
SwissProt	46,287,612	University$^-_\circ$	650

Table 3 illustrates the number (#pairs) of pairs (P,T) such that $P \sqsubseteq T$ with its ratio (%) in all the pairs, and the total and average running time. Note that we just apply the algorithm CATINC to all the pairs without pruning by the number of vertices, and so on.

Table 3 shows that Auction$^-$ has the largest ratio of 13.95% in all the data, and the ratio of Nasa$^-_\circ$ is also more than 10%. One of the reason is that their data are obtained by deleting the root and the caterpillars as the children of the root have similar structures.

On the other hand, Table 3 also shows that SwissProt has the largest average running time in all the data, which is more than twice to the average running time of the other data. Also, $\text{dblp}_{0.1\%}$ has the second largest average running time. One of the reason that the caterpillars in SwissProt and $\text{dblp}_{0.1\%}$ have the larger degree and the larger number of leaves than the other data.

[4]Kyoto Encyclopedia of Genes and Genomes, http://www.kegg.jp/

[5]http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/Software

[6]http://dblp.uni-trier.de/

[7]http://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html

Table 3: The number (#pairs) of pairs (P,T) such that $P \sqsubseteq T$ with its ratio (%) in all the pairs, the total running time (sec) and average running time (msec).

data	#pairs	%	total (sec)	ave. (msec)
N-glycans	8,773	3.33	15.980	0.0603
all-glycans	704,521	1.11	2,252.321	0.0353
CSLOGS	2,070,110	0.12	83,961.780	0.0485
$\text{dblp}_{0.1\%}$	1,855,880	6.99	2,045.844	0.0770
SwissProt	1,400,455	3.03	7,440.401	0.1607
TPC-$\text{H}_\circ$	0	0	0.004	0.0714
Auction^-	9,324	13.95	1.728	0.0259
$\text{Nasa}_\circ^-$	108	10.23	0.030	0.0284
$\text{Protein}_\circ^-$	3,701	0.01	587.739	0.0222
$\text{University}_\circ^-$	1	0.15	0.006	0.0092

Next, we investigate how many caterpillars are included in another caterpillar in the set of caterpillars. Let D be the set of caterpillars. For $P \in D$, we call the number of text caterpillars in D which includes P the *inclusion number* of P in D and denote it by $inc_D(P)$. Furthermore, we define the following formulas.

$$\begin{aligned} h\_inc(k,D) &= |\{P \in D \mid inc_D(P) = k\}|, \\ maxinc(D) &= \max\{inc_D(P) \mid P \in D\}, \\ pat(D) &= \sum_{k=1}^{maxinc(D)} h\_inc(k,D), \\ ratio(D) &= \frac{pat(D)}{|D|}. \end{aligned}$$

Since every pattern caterpillar P having $T \in D$ such that $P \sqsubseteq T$ is counted just once in $pat(D)$ for D, $pat(D)$ is the number of caterpillars in D included in some caterpillar (without themselves) in D. Then, $ratio(D)$ is the ratio of such caterpillars in D.

Tables 4, 5 and 6 illustrate the histograms of $h\_inc(k,D)$ with $maxinc(D)$, $pat(D)$ and $ratio(D)$ for every D. Here, since Auction^-, $\text{Nasa}_\circ^-$ and $\text{University}_\circ^-$ have particular distributions, we summarize them as Table 6.

Tables 4 and 5 (except Table 6) show that SwissProt has the largest value of *maxinc* in all the data. Also, all-glycans has the larger value of *maxinc* than CSLOGS and $\text{dblp}_{0.1\%}$, whereas all-glycans has the much smaller value of "#pairs" in Table 3 than CSLOGS and $\text{dblp}_{0.1\%}$. Furthermore, whereas CSLOGS has the largest value of *pat*, it also has the largest value of "#" in Table 1 and "#pairs" in Table 3. On the other hand, $\text{dblp}_{0.1\%}$ has the largest value of *ratio* and SwissProt has the second largest value of *ratio* over 90%.

Hence, we conjecture that the values of *maxinc* and *ratio* are independent from the value of *inc* and depend on the forms of caterpillars in data.

Table 4: The histograms of $h\_inc(k,D)$ with $maxinc(D)$, $pat(D)$ and $ratio(D)$ (%) for N-glycans, all-glycans and CSLOGS.

N-glycans		all-glycans		CSLOGS	
k	$h\_inc(k,D)$	k	$h\_inc(k,D)$	k	$h\_inc(k,D)$
1	69	1	681	1	4,772
2	33	2	462	2	2,377
3	20	3	370	3	1,524
4	22	4	275	4	1,038
5	11	5	184	5	884
6	16	6	209	6	641
7	16	7	154	7	534
8	16	8	167	8	454
9	8	9	119	9	429
10	9	10	127	10	330
≥ 11	174	≥ 11	4,105	≥ 11	13,215
220	*maxinc*	2,938	*maxinc*	1,702	*maxinc*
394	*pat*	6,853	*pat*	26,198	*pat*
76.65	*ratio* (%)	85.83	*ratio* (%)	62.99	*ratio* (%)

Table 5: The histograms of $h\_inc(k,D)$ with $maxinc(D)$, $pat(D)$ and $ratio(D)$ (%) for $\text{dblp}_{0.1\%}$, SwissProt and $\text{Protein}_\circ^-$.

$\text{dblp}_{0.1\%}$		SwissProt		$\text{Protein}_\circ^-$	
k	$h\_inc(k,D)$	k	$h\_inc(k,D)$	k	$h\_inc(k,D)$
1	42	1	361	1	494
2	38	2	250	2	43
3	32	3	202	3	28
4	29	4	166	4	26
5	34	5	153	5	17
6	26	6	141	6	12
7	27	7	101	7	16
8	26	8	98	8	15
9	25	9	85	9	10
10	19	10	116	10	8
≥ 11	4,794	≥ 11	4,488	≥ 11	93
1,939	*maxinc*	5,666	*maxinc*	149	*maxinc*
5,092	*pat*	6,161	*pat*	762	*pat*
98.80	*ratio* (%)	90.55	*ratio* (%)	14.80	*ratio* (%)

Finally, we give the examples of a pattern caterpillar P and a text caterpillar T such that $inc_D(P) = 1$. for several data D.

Figure 2 illustrates an example of $P = \text{G00954}$ and $T = \text{G04792}$ in N-glycans. Also Figure 3 illustrates an example of $P = \text{G10338}$ and $T = \text{G10334}$ in all-glycans.

On the other hand, Figure 4 illustrates an example of $P = \text{CL33073}$ and $T = \text{CL30743}$ in CSLOGS. Furthermore, Figure 5 illustrates an example of $P =$ ID-T10728_005 and $T =$ ID-T33084_006 in $\text{Protein}_\circ^-$.

Table 6: The histograms of $h\_inc(k,D)$ with $maxinc(D)$, $pat(D)$ and $ratio(D)$ (%) for Auction$^-$, Nasa$^-_\circ$ and University$^-_\circ$.

Auction$^-$

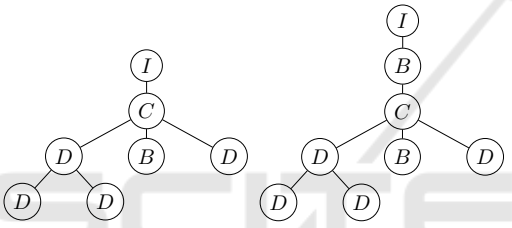
k	$h\_inc(k,D)$
36	259
36	*maxinc*
259	*pat*
100.00	*ratio* (%)

Nasa$^-_\circ$

k	$h\_inc(k,D)$
1	5
2	3
3	3
4	2
5	2
6	2
7	2
8	2
9	2
10	1
10	*maxinc*
24	*pat*
72.73	*ratio* (%)

University$^-_\circ$

k	$h\_inc(k,D)$
1	1
1	*maxinc*
1	*pat*
3.85	*ratio* (%)

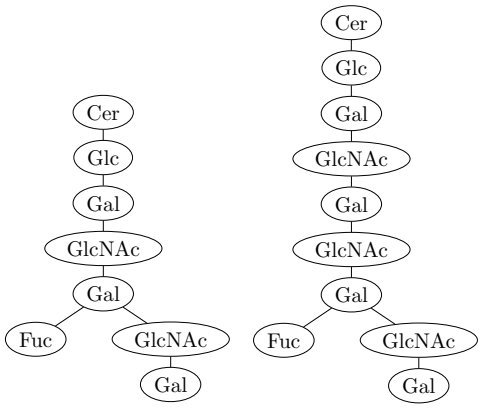


$P = \text{G10338}$ $T = \text{G10334}$

Figure 2: P and T in N-glycans.

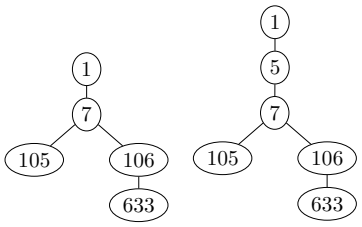
6 CONCLUSION

In this paper, we have designed the algorithm CATINC to determine whether or not $P \sqsubseteq T$ in $O((h+H)\sigma)$ time, where h is the height of P, H is the height of T and σ is the number of labels occurring in P and T. Also, we have given experimental results for the algorithm CATINC by using real caterpillar data.



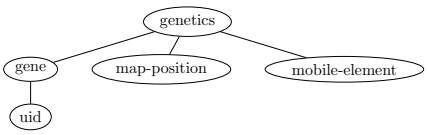
$P = \text{G00057}$ $T = \text{G00073}$

Figure 3: P and T in all-glycans.

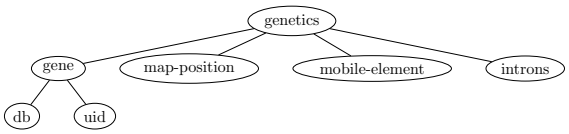


$P = \text{CL33073}$ $T = \text{CL00501}$

Figure 4: P and T in CSLOGS.



$P = \text{ID-T10728\_005}$



$T = \text{ID-T33084\_006}$

Figure 5: P and T in Protein$^-_\circ$.

Note that the algorithm CATINC just determines whether or not $P \sqsubseteq T$ but does not count the number of matching positions when $P \sqsubseteq T$. Then, it is a future work to improve the algorithm CATINC to store the matching positions and count them. Here, it is necessary to count the correspondence between multisets of labels in leaves carefully.

Since the tree inclusion is NP-complete, it is also a future work to introduce the heuristic algorithm by incorporating the algorithm CATINC with the heavy caterpillar of trees such as (Abe et al., 2020), for example.

REFERENCES

Abe, N., Yoshino, T., and Hirata, K. (2020). Heavy caterpillar distance for rooted labeled unordered trees. In *Proc. ICPRAM'20*, pages 198–204.

Akutsu, T., Fukagawa, D., Halldórsson, M. M., Takasu, A., and Tanaka, K. (2013). Approximation and parameterized algorithms for common subtrees and edit distance between unordered trees. *Theoret. Comput. Sci.*, 470:10–22.

Akutsu, T., Jansson, J., Li, R., Takasu, A., and Tamura, T. (2021). New and improved algorithms for unordered tree inclusion. *Theoret. Comput. Sci.*, 883:83–98.

Gallian, J. A. (2007). A dynamic survey of graph labeling. *Electorn. J. Combin.*, 14:DS6.

Hirata, K., Yamamoto, Y., and Kuboyama, T. (2011). Im-

proved MAX SNP-hard results for finding an edit distance between unordered trees. In *Proc. CPM'11 (LNCS 6661)*, pages 402–415.

Hokazono, T., Kan, T., Yamamoto, Y., and Hirata, K. (2012). An isolated-subtree inclusion for unordered trees. In *Proc. IIAI AAI '12*, pages 345–350.

Kilpeläinen, P. and Mannila, H. (1995). Ordered and unordered tree inclusion. *SIAM J. Comput.*, 24:340–356.

Matoušek, J. and Thomas, R. (1992). On the complexity of finding iso- and other morphisms for partial *k*-trees. *Discrete Math.*, 108:343–364.

Muraka, K., Yoshino, T., and Hirata, K. (2018). Computing edit distance between rooted labeled caterpillars. In *Proc. FedCSIS'18*, pages 245–252.

Shamir, R. and Tsur, D. (1999). Faster subtree isomorphism. *Algorithmica*, 33:267–280.

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

Ukita, Y., Yoshino, T., and Hirata, K. (2021). Caterpillar alignment distance for rooted labeled caterpillars: Distance based on alignments required to be caterpillars. In *Recent advance in computational optimization*, pages 111–134.

Valiente, G. (2002). *Algorithms on trees and graphs*. Springer.

Valiente, G. (2005). Constrained tree inclusion. *J. Discrete Algorithms*, 3:431–447.

Zhang, K. and Jiang, T. (1994). Some MAX SNP-hard results concerning unordered labeled trees. *Inform. Process. Lett.*, 49:249–254.

Zhang, K., Wang, J., and Shasha, D. (1996). On the editing distance between undirected acyclic graphs. *Internat. J. Found. Comput. Sci.*, 7:43–58.