Reordering a Tree According to an Order on Its Leaves

Laurent Bulteau
LIGM, Université Gustave Eiffel & CNRS, Champs-sur-Marne, France

Philippe Gambette[1]
LIGM, Université Gustave Eiffel & CNRS, Champs-sur-Marne, France

Olga Seminck
Lattice (Langues, Textes, Traitements informatiques, Cognition),
CNRS & ENS/PSL & Université Sorbonne nouvelle, France

Abstract

In this article, we study two problems consisting in reordering a tree to fit with an order on its leaves provided as input, which were earlier introduced in the context of phylogenetic tree comparison for bioinformatics, OTCM and OTDE. The first problem consists in finding an order which minimizes the number of inversions with an input order on the leaves, while the second one consists in removing the minimum number of leaves from the tree to make it consistent with the input order on the remaining leaves. We show that both problems are NP-complete when the maximum degree is not bounded, as well as a problem on tree alignment, answering two questions opened in 2010 by Henning Fernau, Michael Kaufmann and Mathias Poths. We provide a polynomial-time algorithm for OTDE in the case where the maximum degree is bounded by a constant and an FPT algorithm in a parameter lower than the number of leaves to delete. Our results have practical interest not only for bioinformatics but also for digital humanities to evaluate, for example, the consistency of the dendrogram obtained from a hierarchical clustering algorithm with a chronological ordering of its leaves. We explore the possibilities of practical use of our results both on trees obtained by clustering the literary works of French authors and on simulated data, using implementations of our algorithms in Python.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases tree, clustering, order, permutation, inversions, FPT algorithm, NP-hardness, tree drawing, OTCM, OTDE, TTDE

Digital Object Identifier 10.4230/LIPIcs.CPM.2022.24

Related Version *Previous Version*: `https://hal.archives-ouvertes.fr/hal-03413413v1`

Supplementary Material *Software (Source Code)*: `https://github.com/oseminck/tree_order_evaluation`; archived at `swh:1:dir:84e3aae3efa08e4a2d296abdda6c34da0c6fca6b`

Funding *Philippe Gambette*: "Investissements d'avenir" program, reference ANR-16-IDEX-0003 (I-Site Future, programme "Cité des dames, créatrices dans la cité").
Olga Seminck: "Investissements d'avenir" program, reference ANR19-P3IA-0001 (PRAIRIE 3IA Institute).

1 Introduction

The problem of optimizing the consistency between a tree and a given order on its leaves was first introduced in bioinformatics in the context of visualization of multiple phylogenetic trees in order to highlight common patterns in their subtree structure [6], under the name "one-layer STOP (stratified tree ordering problem)". The authors provided an $O(n^2)$ time

[1] corresponding author

algorithm to minimize, by exchanging the left and right children of internal nodes, the number of inversions between the left-to-right order of the leaves of a binary tree and an input order on its leaves. The problem was renamed OTCM (One-Tree Crossing Minimization) in [9], where an $O(n \log^2 n)$ time algorithm is provided, as well as a reduction to 3-Hitting Set of a variant of the problem where the goal is to minimize the number of leaves to delete from the tree in order to be able to perfectly match the input order on the remaining leaves, called OTDE (One-Tree Drawing by Deleting Edges). An $O(n \log^2 n / \log \log n)$ time algorithm is later provided for OTCM by [1], improved independently in 2010 by [10] and [22] to obtain an $O(n \log n)$ time complexity. About OTDE, the authors of [10] note that "the efficient dynamic-programming algorithm derived for the related problem OTCM [...] cannot be transferred to this problem. However, we have no proof for NP-hardness for OTDE nor TTDE, either". TTDE (Two-Tree Drawing by Deleting Edges) is a variant of OTDE where two leaf-labeled trees are provided as input and the goal is to delete the minimum number of leaves such that the remaining leaves of both trees can be ordered with the same order. We give below an answer to both sentences, providing a dynamic-programming algorithm solving OTDE for trees with fixed maximum degree as well as an NP-hardness proof in the general case for OTDE and for TTDE.

Although this problem was initially introduced in the context of comparing tree embeddings, one tree having its embedding (that is the left-to-right order of all children) fixed, we can note that only the order on the leaves of the tree with fixed embedding is useful to define both problems OTCM and OTDE. Both problems therefore consist not really in comparing trees but rather in reordering the internal nodes of one tree in order to optimize its consistency with an order on its leaves provided as input. A popular problem consisting in finding an optimal order on the leaves of a tree is "seriation", often used for visualization purposes [7], where the optimized criterion is computed on data used to build the tree. For example, a classical criterion, called "optimal leaf ordering", is to maximize the similarity between consecutive elements in the optimal order [2, 3, 4]. Another possibility is to minimize a distance criterion, the "bilateral symmetric distance", computed on pairs of elements in consecutive clusters [5]. Seriation algorithms have been implemented for example in the R-packages `seriation` [12] and `dendsort` [19].

With the OTCM and OTDE problems, our goal is not to reorder a tree using only the original data from which it has been built, but using external data about some expected order on its leaves. In the context where the leaves of the tree can be ordered chronologically, for example, this would help providing an answer to the question: how much is this tree consistent with the chronological order? This issue is relevant for several fields of digital humanities, when objects associated with a publication date are classified with a hierarchical clustering algorithm, for example literature analysis [14], political discourse analysis [15] or language evolution [17], as noticed in [11]. In these articles, the comments about the chronological signal which can be observed in the tree obtained from the clustering algorithm are often unclear or imprecise. For example, in [17], the author observes about Figure 15 on page 17 that "the cluster tree gives a visual representation consistent with what is independently known of the chronological structure of the corpus". However, the structure of the tree does not perfectly reflect the chronology[2]. The algorithms solving the OTCM and OTDE problems can also prevent researchers from claiming having obtained perfect chronological trees with clustering, whereas there are still small inconsistencies that are not easy to spot

[2] For example `1380Gawain.txt` cannot be ordered between `1375AllitMorteArthur.txt` and `1400YorksPlays.txt`.

with the naked eye. For example, although “Chez Jacques Chirac, l’examen des parentés [dans ses discours de vœux] ne suppose aucune rupture, la chronologie étant parfaitement représentée”[3] is claimed about Figure 2.4 in [15], the 1999 speech cannot be ordered between 1998 and 2000.

In this article, we first give useful definitions in Section 1.1. We answer two open problems from [10], proving that OTDE and TTDE are NP-complete, as well as OTCM, in Section 2. We then provide a dynamic programming algorithm solving OTDE in polynomial time for trees with fixed maximum degree in Section 3. This algorithm also works in the more general case where the order on the leaves is not strict. We then provide an FPT algorithm for the OTDE problem parameterized by the deletion-degree of the solution, which is lower than the number of leaves to delete, in Section 4. We also give an example of a tree and an order built to have a distinct solution for the OTCM and OTDE problems in Section 5. Finally, we illustrate the relevance of this problem, and of our implementations of algorithms solving them, for applications in digital humanities, with experiments on trees built from literary works, as well as simulated trees, in Section 6.

1.1 Definitions

Given a set X of elements, we define an X-tree T as a rooted tree whose leaves are bijectively labeled by the elements of X. The set of leaves of T is denoted by $L(T)$ and the set of leaves below some vertex v of T is denoted by $L(T, v)$ (or simply $L(v)$ if T is clear from the context). A set of vertices of T is *independent* if no vertex of T is an ancestor of another vertex of T.

We say that σ is a strict order on X if it is a bijection from X to $[1..n]$ and that it is a weak order on X if it is a surjection from X to $[1..m]$, where $|X| \geq m$. Given any (strict or weak) order σ, we denote by $a \leq_\sigma b$ the fact that $\sigma(a) \leq \sigma(b)$ and by $a <_\sigma b$ the fact that $\sigma(a) < \sigma(b)$. Considering the elements $x_1, \dots, x_n$ of X such that for each $i \in [1..n-1], \sigma(x_i) \leq \sigma(x_{i+1})$, we denote by $(x_1 x_2 \dots x_n)$ the (weak or strict) order σ.

Given an X-tree T and a (weak or strict) order σ on X, we say that an independent pair $\{u, v\}$ of vertices of T is a *conflict wrt.* σ if there exist leaves $a, c \in L(u)$ and $b \in L(v)$ such that $a <_\sigma b <_\sigma c$. Conversely, if $\{u, v\}$ is not a conflict, then either $a \leq_\sigma b$ for all $a \in L(u), b \in L(v)$, or $b \leq_\sigma a$; we then write $u \preceq_\sigma v$ or $v \preceq_\sigma u$, respectively. We say that σ is *suitable* on T if T has no conflict with respect to σ.

Given two (strict or weak) orders σ_1 and σ_2 on X and two elements $a \neq b$ of X, we say that $\{a, b\}$ is an *inversion* for σ_1 and σ_2 if $a \leq_{\sigma_1} b$ and $b <_{\sigma_2} a$, or $b \leq_{\sigma_1} a$ and $a <_{\sigma_2} b$.

Given an X-tree T, a subset X' of X and an order σ on X, we denote by $\sigma[X']$ the order σ restricted to X', and by $T[X']$ the tree T restricted to X', that is the X'-tree obtained from T by removing leaves labeled by $X \setminus X'$ and contracting any arc to a non-labeled leaf, any arc from an out-degree-1 vertex. We define the *deletion-degree* of X' as the maximum degree of the tree induced by the deleted leaves, i.e., $T[X \setminus X']$. Intuitively, the deletion-degree measures how deletions in different branches converge on a few nodes or if they merge progressively. Note that by definition, the deletion-degree of X' is upper-bounded both by the maximum degree of T and by the size of $X \setminus X'$.

We now define the two main problems addressed in this paper (see Figure 1 for an illustration). As explained in the introduction, we differ from previous definitions which considered two trees, one with a fixed order on the leaves, as input, as only the leaf order of the second tree is useful to define the problem and not the tree itself.

[3] “For Jacques Chirac, the examination of the genealogy [of his new year addresses] shows no discontinuity, the chronology being perfectly represented”

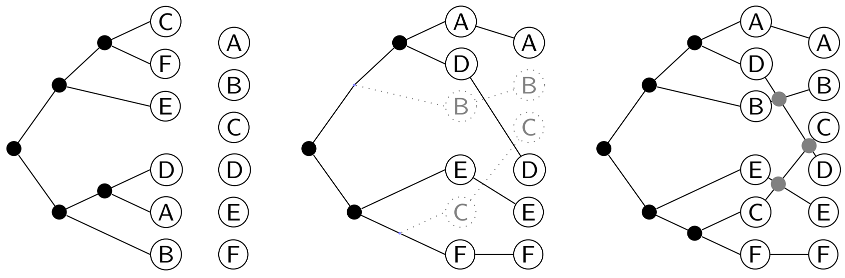

Figure 1 Example for the OTDE and OTCM problem. Left: a tree T on leaves $\{\mathsf{A}, \ldots, \mathsf{F}\}$, the reference permutation is $\sigma = (\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{F})$ (more precisely, $\sigma(\mathsf{A}) = 1, \ldots, \sigma(\mathsf{F}) = 6$). Middle: a solution for OTDE with cost 2. The subtree $T[X']$ for $X' = \{\mathsf{A}, \mathsf{D}, \mathsf{E}, \mathsf{F}\}$ is ordered to show the absence of conflicts with $\sigma[X']$. Right: a solution for OTCM with cost 3. The order $\sigma' = (\mathsf{A}, \mathsf{D}, \mathsf{B}, \mathsf{E}, \mathsf{C}, \mathsf{F})$ is suitable for T and yields three inversions with σ.

We therefore define the OTCM (One-Tree Crossing Minimization) problem as follows:

- **Input**: An X-tree T, an order σ on X and an integer k.
- **Output**: Yes if there exists an order σ' on X suitable on T such that the number of inversions for σ' and σ is at most k, no otherwise.

We also define the OTDE (One-Tree Drawing by Deleting Edges) problem as follows:

- **Input**: An X-tree T, an order σ on X and an integer k.
- **Output**: Yes if there exists a subset X' of X of size at least $|X| - k$ such that $\sigma[X']$ is suitable on $T[X']$, no otherwise.

We finally define the TTDE (Two-Tree Drawing by Deleting Edges) problem in the following way:

- **Input**: Two X-trees T_1 and T_2 and an integer k.
- **Output**: Yes if there exists a subset X' of X of size at least $|X| - k$ and an order σ' on X' that is suitable on $T_1[X']$ and on $T_2[X']$, no otherwise.

2 NP-hardness

2.1 OTDE and TTDE are NP-complete for trees with unbounded degree

▶ **Theorem 1.** *The OTDE problem is NP-complete for strict orders and therefore for weak orders.*

Proof. First note that OTDE is in NP, since, given an X-tree T, an order σ and a set L of leaves to remove, we can check in linear time, by a recursive search of the tree, saving on each node the minimum and the maximum leaf in $\sigma[X - L]$ appearing below, whether $\sigma[X - L]$ is suitable on $T[X - L]$. Regarding NP-hardness, we now give a reduction from Independent Set, which is NP-hard on cubic graphs [16], to OTDE when the input trees have unbounded degree.

We consider an instance of the Independent Set problem, that is a cubic graph $G = (V = \{v_1, \ldots, v_n\}, E)$ such that $|E| = m = 3n/2$ and an integer k. For each vertex v_i, we write e_i^1, e_i^2 and e_i^3 for the three edges incident with v_i (ordered arbitrarily).

We now define an instance of the OTDE problem. The set of leaf labels consists of *vertex labels* denoted v_i and v'_i for each $i \in [1..n]$, one *edge label* for each edge (also denoted e_i^j for the jth edge incident on vertex v_i), and a set of n^2 *separating labels* $B_i = \{b_i^1, b_i^2, \dots b_i^{n^2}\}$ for each $i \in [1..n-1]$.

First, we define the strict order $\sigma(G) = (v_1 e_1^1 e_1^2 e_1^3 v'_1 b_1^1 b_1^2 \dots b_1^{n^2} v_2 e_2^1 e_2^2 e_2^3 v'_2 b_1^2 b_2^2 \dots b_{n-1}^{n^2} v_n e_n^1 e_n^2 e_n^3 v'_n)$. Then, let T_{v_i} be the tree with leaves v_i and v'_i attached below the root, T_e be the tree with leaves $e_i^{i'}$ and $e_j^{j'}$ attached below the root for each edge $e = \{v_i, v_j\}$ of G (with $i', j' \in [1..3]$), and T_{B_i} be the tree with leaves $b_i^1, \dots, b_i^{n^2}$ attached below the root for each $i \in [1..n-1]$. We finally define $T(G)$ as the tree such that $T_{v_1}, T_{v_2}, \dots, T_{v_n}, T_{e_1}, T_{e_2}, \dots T_{e_m}, T_{B_1}, T_{B_2}, \dots$ and $T_{B_{n-1}}$ are attached below the root.

We claim that G has an independent set of size at least $k \Leftrightarrow$ the instance $(T(G), \sigma(G))$ of the OTDE problem has a solution with a set L of at most $m + n - k$ leaves to remove.

$\Rightarrow$: Suppose that there exists a size-k independent set $S = \{s_1, \dots, s_k\}$ of G. We then remove the following leaves (also contracting along the way the edge from their parent to the root of $T(G)$) in order to get a new tree T':

- for each edge $e = \{v_i, v_j\} = e_i^{i'} = e_j^{j'}$ with $i < j$, we remove $e_i^{i'}$ and call $T_{e_j^{j'}} = T_e$ if $v_i \in S$ or if neither v_i nor v_j belong to S; and we remove $e_j^{j'}$ and call $T_{e_i^{i'}} = T_e$ if $v_j \in S$ (as S is an independent set we cannot have both v_i and v_j in S);
- for each vertex v_i not in S we remove v'_i.

By ordering the children of the root of $T(G)$ such as in Figure 2(1), that is by putting, for each v_i with $i \in [1..n]$, T_{v_i}, then $T_{e_i^1}$, $T_{e_i^2}$ and $T_{e_i^3}$ for each of the $e_i^{i'}$ which were not removed and then T_{B_i} (except for $i = n$), the order $\sigma(G)$ restricted to the remaining $m + n + k + n^2(n-1)$ leaves is suitable on T'.

$\Leftarrow$: Suppose that there exists a set L of at most $m + n - k$ leaves such that $\sigma(G)[X - L]$ is suitable on $T(G)[X - L]$. For each parent p_{B_i} of the leaves of B_i and any other vertex v of T such that $\{p_{B_i}, v\}$ is a conflict wrt. $\sigma(G)$, we can delete this conflict either by deleting no leaf of B_i or all leaves of B_i. As each B_i has size $n^2 > m + n - k$, its leaves cannot belong to the set L of leaves to be deleted.

We now consider the trees T_{e_i} for each $i \in [1..m]$: by construction of $\sigma(G)$, as both leaves of each such tree are separated by some $B_{i'}$, therefore by $n^2 > m + n - k$ leaves, one of these two leaves has to be removed, so it has to belong to L. We call L' the set of such leaves of L, therefore there exists a set $L - L'$ of at most $n - k$ other leaves to delete. So there exists a subset S_L of $[1..n]$ of size at least k such that for any element $i \in S_L$, neither v_i, nor v'_i, nor any of the leaves e_i^j for $j \in \{1, 2, 3\}$ belong to $L - L'$. Note that for such $i \in S_L$, all vertices v_i and v'_i are not in L and all e_i^j are in L'. We claim that the vertices of G corresponding to S_L are an independent set of G. Suppose for contradiction that it is not the case, then there exists an edge $e = e_i^{i'} = e_j^{j'}$ between two vertices v_i and v_j of G. By construction of L', exactly one of the leaves labeled by $e_i^{i'}$ and $e_j^{j'}$ is in L' so the second one is in $L - L'$: contradiction. ◀

▶ **Corollary 2.** *The TTDE problem is NP-complete.*

Proof. TTDE is clearly in NP. We prove hardness by reduction from OTDE (see Figure 2(2) for an illustration). Consider an instance (T, σ) of OTDE with σ a strict order on n labels X. Introduce a set Y of n new labels. Build T_1 as a caterpillar with $n + 1$ internal nodes forming a path $r_1, \dots, r_{n+1}$ (with root r_1) and $2n$ leaves where each r_i with $i \le n$ has one leaf attached with label $\sigma^{-1}(i) \in X$ (in the same order), and r_{n+1} has n leaves attached labelled with Y. Build T_2 as a tree, where the root has two children y, t, where y has n children which are leaves labelled with Y, and t is the root of a subtree equal to T.

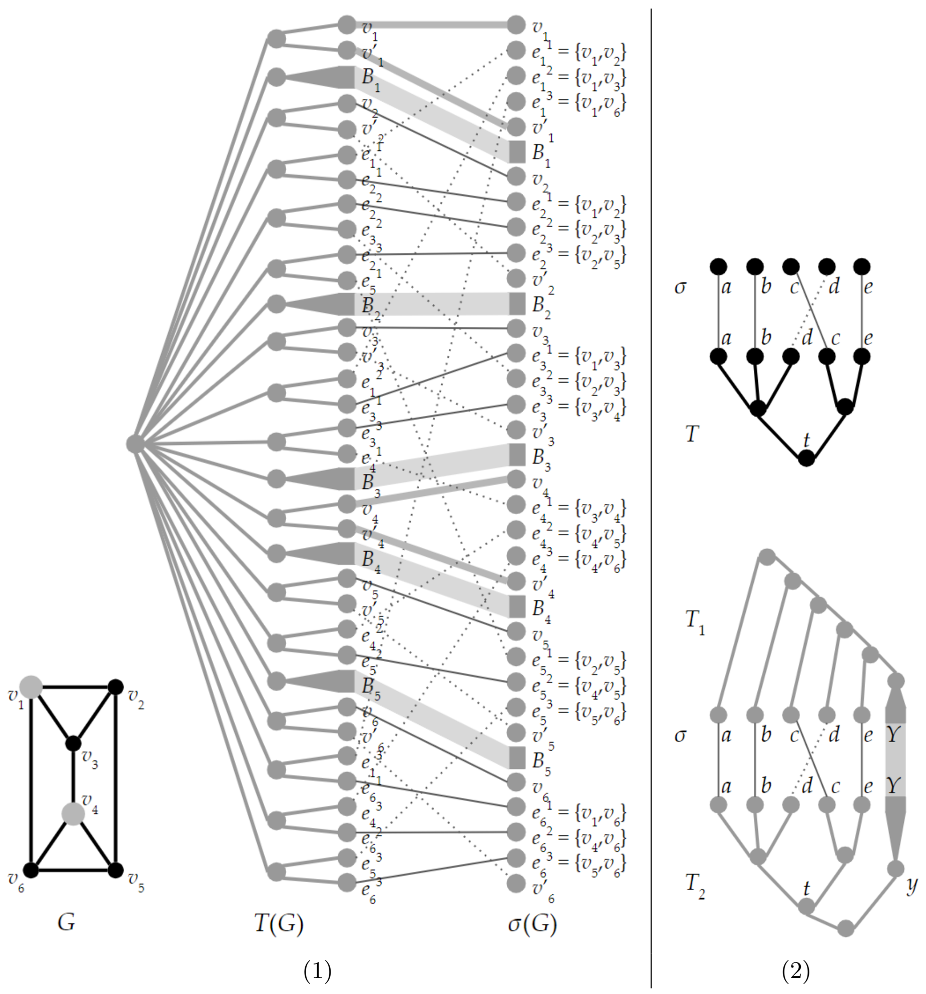

Figure 2 Illustration of the reductions of Independent Set to OTDE and of OTDE to TTDE. (1, left) A graph G with independent set $S = \{v_1, v_4\}$ of size 2. (1, right) The corresponding tree $T(G)$ as well as the order $\sigma(G)$. By removing all leaves connected with dotted lines to the corresponding element in $\sigma(G)$, the resulting subtree of $T(G)$ is suitable for the order (since the remaining arcs are non-crossing). (2) Reduction from an OTDE instance (T, σ) (top) to a TTDE instance (T_1, T_2) (bottom). A large set of leaves labelled Y can be seen as a fixed-point, around which T_1 must be ordered according to σ, and T_2 according to the input tree T.

We now show our main claim: given $0 \leq k < n$, OTDE(T, σ) admits a solution with at most k deletions $\Leftrightarrow$ TTDE(T_1, T_2) admits a solution with at most k deletions.

$\Rightarrow$ Let X' be a size-$(n-k)$ subset of X such that $\sigma[X']$ is suitable on $T[X]$. Then let γ be any order on Y: the concatenation $\sigma[X']\gamma$ is suitable both on $T_1[X' \cup Y]$ and $T_2[X' \cup Y]$, so it is a valid solution for TTDE(T_1, T_2) of size $2n - k$, i.e., with k deletions.

$\Leftarrow$ Let X', Y' be subsets of X, Y, respectively, and σ' be an order on $X' \cup Y'$ such that σ' is suitable on both $T_1[X' \cup Y']$ and $T_2[X' \cup Y']$, and such that $|X' \cup Y'| \geq 2n - k > n$ (in particular, Y' contains at least one element denoted y, and $|X'| \geq n - k$). From T_2, it follows that σ' is the concatenation (in any order) of an order σ_x of X' suitable for $T[X']$ and an order σ_y of Y'. Assume first that σ_x appears before σ_y. Then consider each internal node r_i of the caterpillar T_1 with $i \leq n$ and a child c labelled with an element X'. Then this child must be ordered before all leaves below r_{i+1} since the corresponding subtree contains all leaves labelled with Y. Thus, the nodes in X' are ordered according to $\sigma[X']$, hence $\sigma_x = \sigma[X']$, and $T[X']$ is suitable with $\sigma[X']$. For the other case, where σ_y is ordered before σ_x, then for each r_i with a child in X', this child must be after the subtree with root r_{i+1} (containing Y), and the nodes in X' are ordered according to the reverse of $\sigma[X']$ (i.e., $\sigma_x = \overline{\sigma[X']}$). Thus, the reverse of $\sigma[X']$ is suitable for $T[X']$, and $\sigma[X']$ as well (this is obtained by reversing the permutation of all children of internal nodes of T). In both cases, X' is a solution for OTDE(T, σ) with $|X'| \geq n - k$. ◀

2.2 OTCM is NP-complete for trees with unbounded degree

▶ **Theorem 3.** *The OTCM problem is NP-complete for strict orders and therefore for weak orders.*

Proof. First note that OTCM is in NP, since, given an X-tree T with its leaves ordered according to an order σ' on X suitable on T, an order σ and a set L of leaves, the number of inversions between σ' and σ can be counted in $O(|L|^2)$. Regarding NP-hardness, we now give a reduction from Feedback Arc Set, which is NP-hard [13], to OTCM.

We consider an instance of the Feedback Arc Set problem, that is a directed graph $G = (V = \{v_1, \ldots, v_n\}, A)$ such that $|A| = m$ and an integer f.

We now define an instance of the OTCM problem, illustrated in Figure 3. The set X of leaf labels is $\{v_i^j \mid i \in [1..n], j \in [1..2m]\}$. We define the order $\sigma(G)$ in the following way. For each arc (v_i, v_j) of G, whose rank in the lexicographic order is k, we add to $\sigma(G)$ a k^{th} supplementary ordered sequence (which we will later call a “block” corresponding to this arc) $v_i^{2k-1} v_j^{2k-1} X_{i,j}^{2k-1} \overline{X}_{i,j}^{2k} v_i^{2k} v_j^{2k}$, where $X_{i,j}^{k'}$ is the ordered sequence of $v_{i'}^{k'}$ where i' ranges from 1 to n, excluding i and j, and $\overline{X}_{i,j}^{k'}$ is the reverse of $X_{i,j}^{k'}$ (i.e., the ordered sequence of $v_{i'}^{k'}$ where i' ranges from n down to 1, excluding i and j). The tree $T(G)$ is made of a root with n children v_1 to v_n, each v_i having $2m$ children, the leaves labeled by $v_i^{k'}$ for $k' \in [1..2m]$.

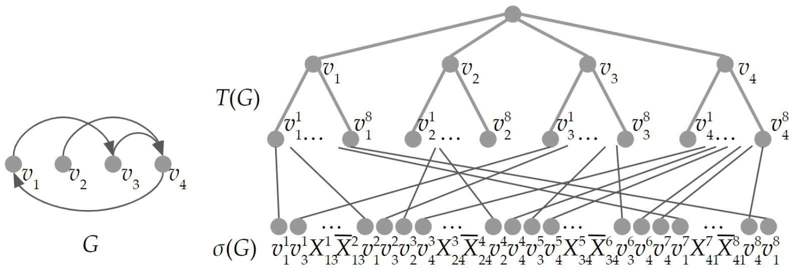
v_1
v_2
v_3
v_4
G
$T(G)$
v_1
v_2
v_3
v_4
$v_1^1 \ldots$
v_1^8
$v_2^1 \ldots$
v_2^8
$v_3^1 \ldots$
v_3^8
$v_4^1 \ldots$
v_4^8
$\sigma(G)$
$v_1^1 v_3^1 X_{13}^1 \overline{X}_{13}^2 v_1^2 v_3^2 v_2^3 v_4^3 X_{24}^3 \overline{X}_{24}^4 v_2^4 v_4^4 v_3^5 v_4^5 X_{34}^5 \overline{X}_{34}^6 v_3^6 v_4^6 v_4^7 v_1^7 X_{41}^7 \overline{X}_{41}^8 v_4^8 v_1^8$

Figure 3 Illustration of the reduction of Feedback Arc Set to OTCM: a graph G with feedback arc set $S = \{(v_4, v_1)\}$ of size 1 and the corresponding tree $T(G)$ as well as the order $\sigma(G)$.

Given an ordering σ' suitable for T, and an inversion $(v_i^k, v_{i'}^{k'})$ forming an inversion between $\sigma(G)$ and σ', we say that this pair is *short-ranged* if $k = k'$, and *long-ranged* otherwise. Furthermore, we say that σ' is *vertex-consistent* if, for every i and $k < k'$, we have $\sigma'(v_i^k) < \sigma(v_i^{k'})$. Finally, given σ', we write σ'' for the permutation of the $[1..n]$ corresponding to the children of the root.

We first claim that for any σ' suitable for T, there are at least $2\binom{n}{2}\binom{2m}{2}$ long-range inversions between σ' and $\sigma(G)$, and this bound is reached if σ' is vertex-consistent. Indeed, pick any pair $(v_i^k, v_{i'}^{k'})$ with $i \neq i'$ and $k \neq k'$. Then $v_i^k <_{\sigma(G)} v_{i'}^{k'}$ iff $k < k'$ (since they are in blocks k and k' of $\sigma(G)$), respectively, and $v_i^k <_{\sigma'} v_{i'}^{k'}$ iff $\sigma''(i) < \sigma''(i')$ (since they are in $L(T, v_i)$ and $L(T, v_{i'})$, respectively). Overall, among $4\binom{n}{2}\binom{2m}{2}$ such pairs of elements, there are $2\binom{n}{2}\binom{2m}{2}$ pairs creating an inversion (which is long-range by definition). For the case $i = i'$, note that pairs $(v_i^k, v_i^{k'})$ do not create any inversion iff σ' is vertex-consistent, which completes the proof of the claim.

Towards counting the number of short-ranged inversions, we say that an arc (v_i, v_j) of G is *satisfied* by σ'' if $\sigma''(i) < \sigma''(j)$. Let $i, j \in [1..n]$ and $k \in [1..m]$, and consider the two pairs (v_i^{2k-1}, v_j^{2k-1}) and (v_i^{2k}, v_j^{2k}). Then these two pairs are, by construction of T, in the same order in σ' (as defined by σ''). If the k^{th} arc of G is (v_i, v_j), then these two pairs are also in the same order in σ, i.e., together they account for either 0 or 2 (short-ranged) inversions. More precisely they yield 0 short-ranged inversions if (v_i, v_j) is satisfied by σ'', and 2 inversions otherwise. If the k^{th} arc of G is any other arc, then exactly one of $(v_i^{2k-1}, v_j^{2k-1}), (v_i^{2k}, v_j^{2k})$ forms a short-ranged inversion. Overall a pair $\{i, j\}$ such that one of $(v_i, v_j), (v_j, v_i)$ is a satisfied arc yields $m-1$ short-ranged inversions, a pair $\{i, j\}$ such that one of $(v_i, v_j), (v_j, v_i)$ is an unsatisfied arc yields $m+1$ short-range inversions, and any other pair $\{i, j\}$ with $i \neq j$ yields m short-ranged inversions. Overall, if there are f unsatisfied arcs, σ' yields $\binom{n}{2}m - m + 2f$ inversions.

We can now complete the proof with our main claim: G has a feedback arc set of size at most f $\Leftrightarrow$ the OTCM problem has a solution with at most $2\binom{n}{2}\binom{2m}{2} + \binom{n}{2}m - m + 2f$ inversions.

$\Rightarrow$: If G has a feedback arc set F of size f, as $G[A-F]$ is acyclic, we consider an order σ'' over n such that for all arcs (v_i, v_j) in $A - F$, $\sigma''(i) < \sigma''(j)$ (i.e., σ'' is the topological order of the vertices in $G[A - F]$). We now order the children v_i of the root of $T(G)$ according to this order σ'' and call σ' the induced order on the leaves of $T(G)$ (also sorting all leaves v_i^j below each v_i by increasing values of j). Note that σ' is vertex-consistent, and that an arc (v_i, v_j) is satisfied by σ'' iff $(v_i, v_j) \notin F$. Thus, σ' yields $2\binom{n}{2}\binom{2m}{2} + \binom{n}{2}m - m + 2f$ inversions.

$\Leftarrow$: Consider an order σ' suitable for T with at most $2\binom{n}{2}\binom{2m}{2} + \binom{n}{2}m - m + 2f$ inversions. Let σ'' be the corresponding order on the leaves of the root, and let F be the set of arcs unsatisfied by σ''. Since σ' has at least $2\binom{n}{2}\binom{2m}{2}$ long-range inversions, it has at most $\binom{n}{2}m - m + 2f$ short-range inversions, and $|F| \leq f$. Finally, since all arcs in $A - F$ are satisfied by σ'', $G[A - F]$ is acyclic and F is a feedback arc set. ◀

3 A polynomial-time algorithm for fixed-degree trees

We start by presenting a dynamic programming algorithm for fixed-degree trees, which is easy to implement and leads to an algorithm in $O(n^4)$ time for binary trees. The FPT algorithm presented in the next section has a better complexity but is more complex and reuses the dynamic programming machinery presented in this section, which explains why we start with this simpler algorithm.

▶ **Theorem 4.** *The OTDE problem can be solved in time $O(d!n^{d+2})$ for trees with fixed maximum degree d and for strict or weak orders.*

Proof. Given a vertex v of a rooted tree T, a (strict or weak) order $\sigma : L(T) \to [1..m]$ and two integers $l \leq r \in [1..m]$. We denote by $\mathcal{X}(v, l, r)$ a subset of $L(T, v)$ of maximum size such that $\sigma[\mathcal{X}(v, l, r)]$ is suitable with $T[\mathcal{X}(v, l, r)]$ and $\forall \ell \in \mathcal{X}(v, l, r), \sigma(\ell) \in [l, r]$. Note that $\mathcal{X}(v, l, r)$ also depends on T and σ but we simplify the notation by not mentioning them as they can clearly be identified from the context.

Denoting by $c_1, \ldots, c_k$ the children of v in T, we claim that the following formula allows to recursively compute $\mathcal{X}(v, l, r)$ in polynomial time:

- $|\mathcal{X}(v, l, r)| = \max\limits_{\substack{\text{permutation } \pi \text{ of } [1..k] \\ x_1 = l \leq x_2 \leq \cdots \leq x_k \leq x_{k+1} = r}} \sum_{i=1}^{k} \left|\mathcal{X}(c_{\pi(i)}, x_i, x_{i+1})\right|$ if v is an internal node of T;
- for any leaf ℓ of T, $|\mathcal{X}(\ell, l, r)| = 1$ if $\sigma(\ell) \in [l, r]$, 0 otherwise.

Correctness. We prove by induction on the size of $L(v)$ that $\mathcal{X}(v,l,r)$ is indeed a subset of $L(T,v)$ of maximum size such that $\sigma[\mathcal{X}(v,l,r)]$ is suitable with $T[\mathcal{X}(v,l,r)]$ and $\forall \ell \in \mathcal{X}(v,l,r), \sigma(\ell) \in [l,r]$.

This is obvious for any leaf, so let us consider a vertex v of T with a set $\{c_1,\dots c_k\}$ of children. Suppose for contradiction that there exists a set of integers $l \le r$ and a subset X' of $L(v)$ of size strictly greater than $\mathcal{X}(v,l,r)$ such that $\sigma[X']$ is suitable with $T[X']$ and $\forall \ell \in X', \sigma(\ell) \in [l,r]$. We then denote by $X'_1,\dots$ and X'_k the sets of leaves $L(c_1) \cap X',\dots$ and $L(c_k) \cap X'$, respectively. Without loss of generality we consider that the children c_i of v are labeled such that $\max_{\ell \in X'_i}\{\sigma(\ell)\} \le \min_{\ell \in X'_{i+1}}\{\sigma(\ell)\}$. For all $i \in [2..k]$, we define $m_i = \min_{\ell \in X'_i}\{\sigma(\ell)\}$, $m_1 = l$ and $m_{k+1} = r$. Using the induction hypothesis we know that for each $i \in [1..k]$, $|X'_i| \le \left|\mathcal{X}(v, \min_{\ell \in X'_i}\{\sigma(\ell)\}, \max_{\ell \in X'_i}\{\sigma(\ell)\})\right|$, so $|X'_i| \le |\mathcal{X}(v,m_i,m_{i+1})|$ because $\left[\min_{\ell \in X'_i}\{\sigma(\ell)\}, \max_{\ell \in X'_i}\{\sigma(\ell)\}\right] \subseteq [m_i,m_{i+1}]$. Therefore, $|X'| = \sum_{i=1}^{k} |X'_i| \le \sum_{i=1}^{k} |\mathcal{X}(v,m_i,m_{i+1})|$ so by definition of $\sigma[\mathcal{X}(v,l,r)]$, $|X'| \le \sigma[\mathcal{X}(v,l,r)]$: contradiction!

We therefore obtain a correct solution of $OTDE(T,\sigma)$ by computing $\mathcal{X}(\text{root}(T),0,m)$.

Running-time. For each v, we compute the table of the $O(n^2)$ values of $\mathcal{X}(v,l,r)$ for all intervals $[l,r]$. Each of these values can be computed by generating the $k!$ permutations of children of v to consider any possible order among the children and splitting the interval $[l,r]$ into any possible configurations of d consecutive intervals with integer bounds partitioning $[l,r]$, which can be done in time $O(n^{d-1})$. So the computation of each $\mathcal{X}(v,l,r)$ is done in time $O(d!n^{d-1})$, therefore the total computation of all $\mathcal{X}(v,l,r)$ is done in time $O(n \times n^2 \times d!n^{d-1})$, that is in $O(d!n^{d+2})$. ◀

4 An FPT algorithm for the *deletion-degree* parameter for OTDE

We recall that with a reduction of OTDE to 3-Hitting Set [10], using the best algorithm known so far to solve this problem[4], we can obtain an algorithm to solve OTDE $O^*(2.08^k)$ [23], where k is the number of leaves to delete and the O^* notation ignores the polynomial factor. In this section we obtain an FPT algorithm in time $O(n^4 d\partial 2^\partial)$, where d is the maximum degree of the tree and ∂ is the deletion-degree of the solution.

▶ **Theorem 5.** *The OTDE problem parameterized by the deletion-degree ∂ of the solution is FPT and can be solved in time $O(n^4 d\partial 2^\partial)$ for strict or weak orders.*

We adapt the dynamic programming algorithm from Theorem 4, using a vertex cover subroutine to have a good estimation of the permutation of the children of each node.

We first introduce some definitions (see Figure 4 for a illustration of these definitions and the algorithm in general). Given any vertex v of T, let C_v be the (independent) set of children of v, and let G_v be the *conflict graph* with vertex set C_v and with one edge per conflict. Let K be a vertex cover of G_v. Then the vertices of $C_v \setminus K$ have a *canonical order* $(w_1,\dots,w_{k'})$, with $k' = |C_v \setminus K|$ and $w_i \preceq_\sigma w_j$ for all $i \le j$ (ties may happen when two children contain a single leaf each which are equal, such ties are broken arbitrarily). We say that $P \subseteq C_v$ is a *prefix of C_v wrt. K* if $P \setminus K$ is a prefix of this order (i.e., for some $i \le k'$, $P \setminus K = \{w_1,\dots,w_i\}$). In other words, ignoring all subtrees below vertices of K, all leaves below vertices of a prefix P are necessarily ordered before leaves below vertices outside of P.

[4] `http://fpt.wikidot.com/fpt-races`

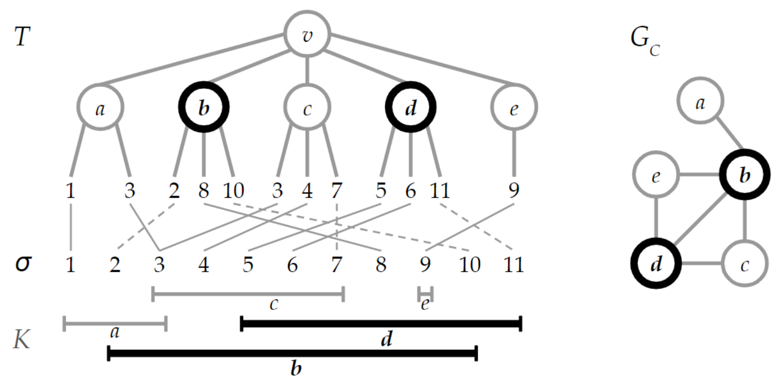

Figure 4 An instance (T, σ) of OTDE (top-left), with a vertex v having children set $C_v = \{a, b, c, d, e\}$. The conflict graph of C_v (right) has a size-2 vertex cover $K = \{b, d\}$. Based on the span of each vertex (bottom-right), the dynamic programming algorithm tests permutations of C_v such that (a, c, e) appear in this order, interleaved in any possible way with b and d. In particular, the final solution corresponds to the permutation $(a\ c\ d\ b\ e)$ of C_v. Note that since σ may be a weak order (two leaves are labelled 3 in the example), the conflict graph does not correspond exactly to the intersection graph of the span intervals, e.g. vertices a and c are *not* in conflict, even though their spans overlap.

▶ **Lemma 6.** *If X' is a solution of OTDE with deletion-degree ∂, then for any vertex v of T, the conflict graph G_v admits a vertex cover of size at most ∂.*

Proof. Given a subset X' of X, we say that a node v of T *has a deletion* if some $L(v) \not\subseteq X'$, i.e., if v has a leaf in $X \setminus X'$. Let $\{u, v\}$ be any conflict (edge) of the conflict graph G_v, then at least one of u, v has a deletion for X' (indeed, the conflict involves three leaves a, b, c, of which at least one must be deleted). Thus, the vertices with a deletion in G_v form a vertex cover of this graph. The lemma follows from the fact that at most ∂ vertices have a deletion in each conflict graph. ◀

The first step of our algorithm consists in computing, for each node v of the graph, the set C of children of v, its conflict graph G_v, and a minimum vertex cover K_v of G_C. Since each K_v has size at most ∂ (by Lemma 6), K_v can be computed in time $O(1.3^{\partial} + \partial n)$ [5], and overall this first step takes $O(1.3^{\partial} n + \partial n^2)$.

We proceed with the dynamic programming part of our algorithm. To this end, we generalize the table $\mathcal{X}$ to sets of nodes (instead of only v) as follows: $\mathcal{X}(P, l, r)$ corresponds to the largest set X of leaves in $\bigcup_{u \in P} L(u)$ such that σ_X is suitable for $T[X]$. Note that for a node v with children set C, $\mathcal{X}(v, l, r) = \mathcal{X}(\{v\}, l, r) = \mathcal{X}(C, l, r)$.

We first compute $\mathcal{X}(\{v\}, l, r)$ for each leaf v: clearly $\mathcal{X}(\{v\}, l, r) = \{u\}$ if $l \leq \sigma(v) \leq r$, and $\mathcal{X}(\{v\}, l, r) = \emptyset$ otherwise. For each internal vertex v (visiting the tree bottom-up), we obtain $\mathcal{X}(\{v\}, l, r)$ by first computing $\mathcal{X}(P, l, r)$ for each prefix P of C_v by increasing order of size, using the following formulas:

$$\begin{aligned} |\mathcal{X}(P, l, r)| &= \emptyset \text{ if } P = \emptyset \\ &= \max_{\substack{x \in [l..r],\ u \in P \\ P \setminus \{u\} \text{ prefix of } C_v}} |\mathcal{X}(P \setminus \{u\}, l, x)| + |\mathcal{X}(\{u\}, x, r)| \\ |\mathcal{X}(\{v\}, l, r)| &= |\mathcal{X}(C_v, l, r)|. \end{aligned}$$

Each vertex v has at most $d2^{\partial}$ prefixes, so the dynamic programming table $\mathcal{X}$ has at most n^3d2^{∂} cells to fill. For each prefix P, there exist at most $\partial + 1$ vertices $u \in P$ such that $P \setminus \{u\}$ is a prefix (u can be any vertex in $P \cap K_v$, or the maximum vertex for $\preceq_\sigma$ in $P \setminus K_v$). Overall, the *max* is taken over $O(n\partial)$ elements, and $\mathcal{X}$ can be filled in time $O(n^4 d\partial 2^{\partial})$.

Before proving the correctness of the above formula, we need a final definition: given a set of leaves $X' \subseteq X$ and a vertex v of T, we write $\text{span}_{X'}(v)$ for the smallest interval containing $\sigma(u)$ for each leaf $u \in L(u) \cap X'$ (note that $\text{span}_{X'}(v)$ may be empty, if all its leaves are deleted in X').

▶ **Lemma 7.** *Let X' be a solution of OTDE(T, σ), $v \in T$ and $1 \leq l \leq r \leq m$ such that $\text{span}_{X'}(v) \subseteq [l, r]$. Then there exists a permutation $(c_1 \ \dots \ c_k)$ of the children of v and integers $x_0 = l \leq x_1 \leq \ldots \leq x_k = r$ such that, for each $i \leq k$,*

(a) *$\text{span}_{X'}(c_i) \subseteq [x_{i-1}, x_i]$, and*

(b) *$P_i = \{c_1, \ldots, c_i\}$ is a prefix of the children of v wrt. σ.*

Proof. Recall that we write C_v and K_v, respectively, for the set of chidren of v and the vertex cover in the conflict graph induced by these children. For each element c of C_v with a non-empty span, let $x(c) = \max(\text{span}(c))$. For each element w_i of $C_v \setminus K_v$ with an empty span (taking i for the rank according to the canonical order), let $x(w_i) = x(w_{i-1})$ (and $x(w_1) = l$ for $i = 1$). For the remaining vertices (in K_v with an empty span), set $x(c) = l$. Finally, order vertices $c_1, \ldots, c_k$ by increasing values of $x(c_i)$ (breaking ties according to the canonical order when applicable, or arbitrarily otherwise), and set $x_i = x(c_i)$.

Condition (a) follows from the fact that X' is a solution for OTDE(T, σ), so that the span covered by the leaves under siblings do not overlap. For condition (b) we refer to the definition of prefix: each $P_i \setminus K_c$ is indeed a prefix in the canonical ordering of $C_v \setminus K_v$. ◀

The dynamic programming formula follows from the above remark: one can build the solution by incrementing prefixes one vertex at a time (rather than trying all possible permutations of children, as in Theorem 4).

5 Optimizing OTCM and OTDE are two different things

In order to ensure that finding the smallest k such that OTCM or OTDE outputs a positive answer actually consists in optimizing different criteria, we provide in Figure 5 an example of X-tree and an order of its leaves where the order reaching the best k for a positive answer of the OTCM problem does not provide the optimal value for the number of leaves to delete in a positive answer of OTDE and where the best k for a positive answer of the OTDE problem does not provide an optimal value for the number of inversions for a positive answer of the OTCM problem.

We checked the optimality for both criteria by implementing the "naive" dynamic programming $O(n^2)$ algorithm described in Section 2.1 of [10] to solve the OTCM problem and the $O(n^4)$ algorithm described in Section 3 to solve the OTDE problem on binary trees. Both implementations are available in Python, under the GPLv3 licence, at `https://github.com/oseminck/tree_order_evaluation`, as well as the file `inputCounterExample1b.txt` containing the Newick encoding for the tree of Figure 5.

6 Experiments and discussion

In this section, we investigate the potential for use of OTCM and OTDE in applications where the tree of elements is obtained from a clustering algorithm taking as input distances between those elements, and where we want to test whether this clustering reflects some

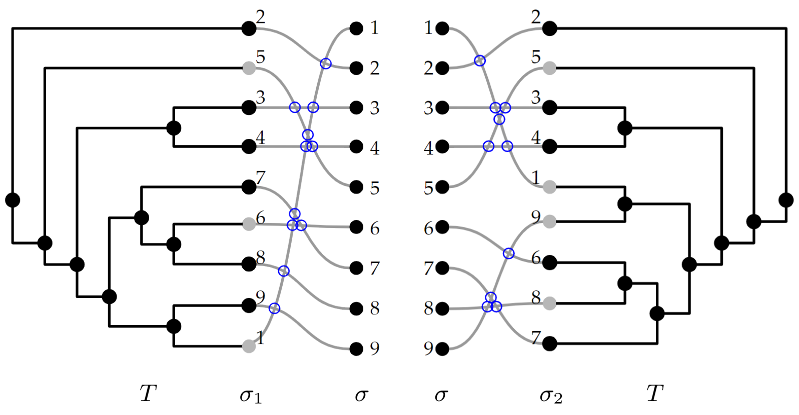

Figure 5 Two planar embeddings of a rooted tree T: the one on the left is optimal for the OTDE problem (deleting the 3 gray leaves makes the order σ suitable on T restricted to the remaining leaves, but the order σ_1 suitable on T has 11 inversions, shown with empty circles, with σ); the other one is optimal for the OTCM problem with the order σ_2 suitable on T having 10 inversions with σ but not for the OTDE problem (4 leaves, for example the 4 gray ones, need to be deleted to make the order σ suitable on T restricted to the remaining leaves).

intrinsic order on the elements, for example the chronological order. We both test the running time of OTCM and OTDE on real data, and the performance of OTDE on simulated data to detect possibly misplaced leaves in the order.

The first experiment deals with text data: the CIDRE corpus [20] that contains the works of 11 French 19$^{\text{th}}$ century fiction writers dated by year (every file contains a book that is annotated with its year of writing). We apply apply hierarchical clustering on the different corpora using the `AgglomerativeClustering` class from the package `sklearn` [18]. Distance matrices on which the clustering is based are obtained by using the relative frequencies of the 500 most frequent tokens[5] in each corpus. Distance matrices were generated using the R package stylo [8], with the *canberra* distance metric. We obtain the results given in Table 1, which provides the running time in milliseconds of the algorithms we implemented to solve OTCM and OTDE. They show that both algorithms on binary trees are quick enough to handle typical instances of the OTCM and the OTDE problems relevant for digital humanities, a few milliseconds for the first one and a few seconds for the second one, for instances of about 50 elements in the tree and in the order.

Investigating precisely whether the numbers of inversions or deleted leaves shown in Table 1 are sufficiently small to reflect consistency with a chronological signal is beyond the scope of this paper. However, we also provide p_{OTCM} and p_{OTDE}, the percentage of cases when the best order on the leaves of the tree has the same number of inversions, or less than the chronological order, among 10000 randomly generated orders for OTCM and 100 randomly generated orders for OTDE, respectively[6]. These numbers illustrate that in all cases, it is unlikely that the observed optimal numbers of inversions or deleted leaves are due

[5] A token is (a part of) a word form or a punctuation marker. The last sentence would yield the following tokens: [“A”, “token”, “is”, “(”, “a”, “part”, “of”, “)”, “a”, “word”, “form”, “or”, “a”, “punctuation”, “marker”, “.”] Deliberately, we do not use the term “word”, because the word can be seen as a linguistic unit of form and meaning, and henceforward “punctuation marker” would be one word and the period in the end of the sentence would not be one.

[6] We chose to generate less random orders for OTDE in our simulations, as our algorithm is slower to solve this problem than OTCM.

Table 1 Results of our implementations for problems OTCM and OTDE on binary trees generated from corpora of French novels of the 19th century. Time durations are given in milliseconds.

tree	**# leaves**	**OTCM time**	**# inversions**	p_{OTCM}	**OTDE time**	**# deleted leaves**	p_{OTDE}
Ségur	22	1	40	0.24	200	9	1
Féval	23	2	47	0.38	268	8	0
Aimard	24	1	35	0	401	8	0
Lesueur	31	1	48	0	676	13	0
Zévaco	29	1	42	0	727	11	0
Zola	35	2	60	0	1203	9	0
Gréville	36	2	105	0	2211	18	1
Ponson	42	3	167	2.23	3447	18	0
Balzac	59	4	248	0	8292	34	0
Verne	58	3	183	0	13446	27	0
Sand	62	4	283	0	17557	39	1

to chance, as we get equal or smaller values of inversions or deleted leaves on less than 3% of random orders (for Ponson du Terrail the number of inversions is 167 or less for 2.23% of random orders; for one of the 10 000 simulated random orders, it reached as little as 124 inversions). These preliminary results obtained thanks to reasonably small running times open new perspectives in investigating further the practical use of these algorithms, and comparing their results with other methods to search for signals of chronological evolution in textual data [21].

Our second experiment involves simulated data, to check whether, in the case the tree is built to be consistent with the input order, our algorithm finding the minimum of leaves in the tree to remove inconsistencies with the order is able to detect errors that we intentionally add to the order. We produced 100 instances of the OTDE problem, for each chosen value of n, the number of leaves, and $e < n$, the number of errors, in the following manner:

1. we randomly pick n distinct integers from the interval $[0, 999]$, which will be our set X of leaves;
2. we build a distance matrix in which the distance between two elements from X is simply the absolute difference between both; we add some noise to this matrix by adding or subtracting in each cell a random quantity equal to at most 10% of the cell value, obtaining a noisy matrix, from which we build an X-tree T using the `AgglomerativeClustering` class from the package `sklearn`;
3. we randomly pick a set L_e of e leaves in X and replace their value by another integer, randomly chosen from the interval $[0, 999]$, distinct from other leaf labels; σ is the set of leaves ordered by increasing value taking into account these new values;
4. by solving the OTDE problem on T and σ, we compute the minimum set L of leaves to remove to make $\sigma[X - L]$ suitable on $T[X - L]$, and check whether $L = L_e$.

This experiment simulates the situation where we would have dating errors on the elements we clustered in a tree. Note that like in the case of dating errors, the error in our simulation may not change the overall order on the leaves. Table 2 provides, for each chosen values of n and e, the proportion of simulated instances of $OTDE$ where $L = L_e$, that is when our algorithm removed exactly the e leaves whose label had been randomly modified. We can observe that this happens in a majority of cases only when the number of modified leaves is small compared with the total number of leaves (up to 2 for 20 leaves, up to 4 for 50 leaves). Solving OTDE still allows to identify $e - 1$ among the e modified leaves in a majority of cases in all our experiments.

Table 2 Results of the attempts to perfectly detect the set L_e of randomly relabeled leaves in simulated trees (when $L = L_e$); the situation when $|L - L_e| = 1$ corresponds to finding only $e - 1$ leaves among the e randomly relabeled leaves).

n = # leaves	e = # errors	proportion of cases when $L = L_e$	when $\|L - L_e\| = 1$
20	1	0.79	1
20	2	0.62	0.96
20	3	0.39	0.88
20	4	0.33	0.77
20	5	0.27	0.67
50	1	0.93	1
50	2	0.83	0.99
50	3	0.70	0.98
50	4	0.59	0.91
50	5	0.56	0.90

7 Conclusion and perspectives

In this article, we addressed two problems initially introduced with motivations from bioinformatics, OTCM and OTDE. We stated them in a more simple framework with a tree and an order as input, instead of two trees as was the case when they were introduced, opening perspectives for new practical uses in digital humanities and proving that they are not equivalent. We proved that both problems, as well as a problem on two trees, TTDE, are NP-complete in the general case. We gave a polynomial-time algorithm for OTDE on trees with fixed maximum degree and an FPT algorithm in a parameter possibly smaller than the size of the solution for arbitrary trees.

We also investigated their potential for practical use, checking that the algorithms we implemented with open source code in Python to solve them are well suited for applications in digital humanities in terms of running time. We also observed on simulated data that it is possible to identify a small number of leaves for which there would be an ordering error if the tree is built from distance data derived from an order on its leaves. Future research includes the search for FPT algorithms, with relevant parameters, for OTCM and TTDE.

References

1 Mukul S Bansal, Wen-Chieh Chang, Oliver Eulenstein, and David Fernández-Baca. Generalized binary tanglegrams: Algorithms and applications. In *International Conference on Bioinformatics and Computational Biology*, pages 114–125. Springer, 2009. doi:10.1007/978-3-642-00727-9_13.

2 Ziv Bar-Joseph, Erik D. Demaine, David K. Gifford, Nathan Srebro, Angèle M. Hamel, and Tommi S. Jaakkola. K-ary clustering with optimal leaf ordering for gene expression data. *Bioinformatics*, 19(9):1070–1078, 2003. doi:10.1093/bioinformatics/btg030.

3 Ziv Bar-Joseph, David K Gifford, and Tommi S Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(suppl 1):S22–S29, 2001. doi:10.1093/bioinformatics/17.suppl_1.S22.

4 Ulrik Brandes. Optimal leaf ordering of complete binary trees. *Journal of Discrete Algorithms*, 5(3):546–552, 2007. doi:10.1016/j.jda.2006.09.003.

5 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.

6 Tim Dwyer and Falk Schreiber. Optimal leaf ordering for two and a half dimensional phylogenetic tree visualisation. In *APVis '04: Proceedings of the 2004 Australasian symposium on Information Visualisation*, volume 35, pages 109–115, 2004. doi:10.5555/1082101.1082114.

7 Denise Earle and Catherine B. Hurley. Advances in dendrogram seriation for application to visualization. *Journal of Computational and Graphical Statistics*, 24(1):1–25, 2015. doi:10.1080/10618600.2013.874295.

8 Maciej Eder, Jan Rybicki, and Mike Kestemont. Stylometry with R: a package for computational text analysis. *R Journal*, 8(1):107–121, 2016. URL: https://journal.r-project.org/archive/2016/RJ-2016-007/index.html.

9 Henning Fernau, Michael Kaufmann, and Mathias Poths. Comparing trees via crossing minimization. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 457–469. Springer, 2005. doi:10.1007/11590156_37.

10 Henning Fernau, Michael Kaufmann, and Mathias Poths. Comparing trees via crossing minimization. *Journal of Computer and System Sciences*, 76(7):593–608, 2010. doi:10.1016/j.jcss.2009.10.014.

11 Philippe Gambette, Olga Seminck, Dominique Legallois, and Thierry Poibeau. Evaluating hierarchical clustering methods for corpora with chronological order. In *EADH2021: Interdisciplinary Perspectives on Data. Second International Conference of the European Association for Digital Humanities*, Krasnoyarsk, Russia, September 2021. EADH. URL: https://hal.archives-ouvertes.fr/hal-03341803.

12 Michael Hahsler, Kurt Hornik, and Christian Buchta. Getting things in order: an introduction to the R package seriation. *Journal of Statistical Software*, 25(3):1–34, 2008. doi:10.18637/jss.v025.i03.

13 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

14 Cyril Labbé and Dominique Labbé. Existe-t-il un genre épistolaire? Hugo, Flaubert et Maupassant. In *Nouvelles Journées de l'ERLA*, pages 53–85. L'Harmattan, 2013.

15 Jean-Marc Leblanc. *Analyses lexicométriques des vœux présidentiels*. ISTE Group, 2016.

16 Bojan Mohar. Face covers and the genus problem for apex graphs. *Journal of Combinatorial Theory, Series B*, 82(1):102–117, 2001. doi:10.1006/jctb.2000.2026.

17 Hermann Moisl. How to visualize high-dimensional data: a roadmap. *Journal of Data Mining & Digital Humanities*, 2020. doi:10.46298/jdmdh.5594.

18 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

19 Ryo Sakai, Raf Winand, Toni Verbeiren, Andrew Vande Moere, and Jan Aerts. dendsort: modular leaf ordering methods for dendrogram representations in R. *F1000Research*, 3, 2014. doi:10.12688/f1000research.4784.1.

20 Olga Seminck, Philippe Gambette, Dominique Legallois, and Thierry Poibeau. The corpus for idiolectal research (CIDRE). *Journal of Open Humanities Data*, 7:15, 2021. doi:10.5334/johd.42.

21 Olga Seminck, Philippe Gambette, Dominique Legallois, and Thierry Poibeau. The evolution of the idiolect over the lifetime: A quantitative and qualitative study on French 19th century literature, 2022. Under review.

22 Balaji Venkatachalam, Jim Apple, Katherine St John, and Dan Gusfield. Untangling tanglegrams: comparing trees by their drawings. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(4):588–597, 2010. doi:10.1109/TCBB.2010.57.

23 Magnus Wahlström. *Algorithms, measures and upper bounds for satisfiability and related problems*. PhD thesis, Department of Computer and Information Science, Linköpings universitet, 2007. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-8714.