



**HAL**  
open science

## Reordering a tree according to an order on its leaves

Laurent Bulteau, Philippe Gambette, Olga Seminck

► **To cite this version:**

Laurent Bulteau, Philippe Gambette, Olga Seminck. Reordering a tree according to an order on its leaves. CPM 2022, Jun 2022, Prague, Czech Republic. pp.24:1-24:15, 10.4230/LIPIcs.CPM.2022.24 . hal-03413413v2

**HAL Id: hal-03413413**

**<https://hal.science/hal-03413413v2>**

Submitted on 15 Apr 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# 1 Reordering a tree according to an order on its 2 leaves

3 **Laurent Bulteau** ✉ 

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

5 **Philippe Gambette**<sup>1</sup> ✉ 

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

7 **Olga Seminck** ✉ 

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

---

## 10 Abstract

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

26 **2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact  
27 algorithms

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

30 **Digital Object Identifier** 10.4230/LIPIcs.CVIT.2016.23

31 **Supplementary Material** [https://github.com/oseminck/tree\\_order\\_evaluation](https://github.com/oseminck/tree_order_evaluation)

32 **Funding** *Philippe Gambette*: “Investissements d’avenir” program, reference ANR-16-IDEX-0003  
33 (I-Site Future, programme “Cité des dames, créatrices dans la cité”).

34 *Olga Seminck*: “Investissements d’avenir” program, reference ANR19-P3IA-0001 (PRAIRIE 3IA  
35 Institute)

---

<sup>1</sup> corresponding author



36 **1** Introduction

37 The problem of optimizing the consistency between a tree and a given order on its leaves  
 38 was first introduced in bioinformatics in the context of visualization of multiple phylogenetic  
 39 trees in order to highlight common patterns in their subtree structure [6], under the name  
 40 “one-layer STOP (stratified tree ordering problem)”. The authors provided an  $O(n^2)$  time  
 41 algorithm to minimize, by exchanging the left and right children of internal nodes, the number  
 42 of inversions between the left-to-right order of the leaves of a binary tree and an input order  
 43 on its leaves. The problem was renamed OTCM (ONE-TREE CROSSING MINIMIZATION)  
 44 in [9], where an  $O(n \log^2 n)$  time algorithm is provided, as well as a reduction to 3-HITTING  
 45 SET of a variant of the problem where the goal is to minimize the number of leaves to delete  
 46 from the tree in order to be able to perfectly match the input order on the remaining leaves,  
 47 called OTDE (ONE-TREE DRAWING BY DELETING EDGES). An  $O(n \log^2 n / \log \log n)$  time  
 48 algorithm is later provided for OTCM by [1], improved independently in 2010 by [10] and [22]  
 49 to obtain an  $O(n \log n)$  time complexity. About OTDE, the authors of [10] note that “the  
 50 efficient dynamic-programming algorithm derived for the related problem OTCM [...] cannot  
 51 be transferred to this problem. However, we have no proof for NP-hardness for OTDE nor  
 52 TTDE, either”. TTDE (TWO-TREE DRAWING BY DELETING EDGES) is a variant of OTDE  
 53 where two leaf-labeled trees are provided as input and the goal is to delete the minimum  
 54 number of leaves such that the remaining leaves of both trees can be ordered with the  
 55 same order. We give below an answer to both sentences, providing a dynamic-programming  
 56 algorithm solving OTDE for trees with fixed maximum degree as well as an NP-hardness  
 57 proof in the general case for OTDE and for TTDE.

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

71 With the OTCM and OTDE problems, our goal is not to reorder a tree using only the  
 72 original data from which it has been built, but using external data about some expected order  
 73 on its leaves. In the context where the leaves of the tree can be ordered chronologically, for  
 74 example, this would help providing an answer to the question: how much is this tree consistent  
 75 with the chronological order? This issue is relevant for several fields of digital humanities,  
 76 when objects associated with a publication date are classified with a hierarchical clustering  
 77 algorithm, for example literature analysis [14], political discourse analysis [15] or language  
 78 evolution [17], as noticed in [11]. In these articles, the comments about the chronological  
 79 signal which can be observed in the tree obtained from the clustering algorithm are often  
 80 unclear or imprecise. For example, in [17], the author observes about Figure 15 on page 17  
 81 that “the cluster tree gives a visual representation consistent with what is independently  
 82 known of the chronological structure of the corpus”. However, the structure of the tree

83 does not perfectly reflect the chronology<sup>2</sup>. The algorithms solving the OTCM and OTDE  
 84 problems can also prevent researchers from claiming having obtained perfect chronological  
 85 trees with clustering, whereas there are still small inconsistencies that are not easy to spot  
 86 with the naked eye. For example, although “*Chez Jacques Chirac, l’examen des parentés*  
 87 *[dans ses discours de vœux] ne suppose aucune rupture, la chronologie étant parfaitement*  
 88 *représentée*”<sup>3</sup> is claimed about Figure 2.4 in [15], the 1999 speech cannot be ordered between  
 89 1998 and 2000.

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

## 101 1.1 Definitions

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

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

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

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

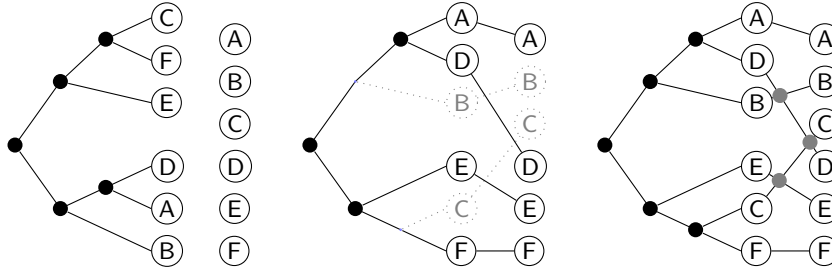
118 Given an  $X$ -tree  $T$ , a subset  $X'$  of  $X$  and an order  $\sigma$  on  $X$ , we denote by  $\sigma[X']$  the order  
 119  $\sigma$  restricted to  $X'$ , and by  $T[X']$  the tree  $T$  restricted to  $X'$ , that is the  $X'$ -tree obtained from  
 120  $T$  by removing leaves labeled by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

<sup>2</sup> For example `1380Gawain.txt` cannot be ordered between `1375AllitMorteArthur.txt` and `1400YorksPlays.txt`.

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

## 23:4 Reordering a tree according to an order on its leaves

124 progressively. Note that by definition, the deletion-degree of  $X'$  is upper-bounded both by  
 125 the maximum degree of  $T$  and by the size of  $X \setminus X'$ .



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

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

130 We therefore define the OTCM (ONE-TREE CROSSING MINIMIZATION) problem as  
 131 follows:

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

135 We also define the OTDE (ONE-TREE DRAWING BY DELETING EDGES) problem as  
 136 follows:

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

140 We finally define the TTDE (TWO-TREE DRAWING BY DELETING EDGES) problem in  
 141 the following way:

- 142 ■ **Input:** Two  $X$ -trees  $T_1$  and  $T_2$  and an integer  $k$ .
- 143 ■ **Output:** Yes if there exists a subset  $X'$  of  $X$  of size at least  $|X| - k$  and an order  $\sigma'$  on  
 144  $X'$  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

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

150 **Proof.** First note that OTDE is in NP, since, given an  $X$ -tree  $T$ , an order  $\sigma$  and a set  $L$   
 151 of leaves to remove, we can check in linear time, by a recursive search of the tree, saving  
 152 on each node the minimum and the maximum leaf in  $\sigma[X - L]$  appearing below, whether

153  $\sigma[X - L]$  is suitable on  $T[X - L]$ . Regarding NP-hardness, we now give a reduction from  
 154 INDEPENDENT SET, which is NP-hard on cubic graphs [16], to OTDE when the input trees  
 155 have unbounded degree.

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

159 We now define an instance of the OTDE problem. The set of leaf labels consists of *vertex*  
 160 *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  
 161 the  $j$ th 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  
 162 each  $i \in [1..n - 1]$ .

163 First, we define the strict order  $\sigma(G) = (v_1 e_1^1 e_1^2 e_1^3 v'_1 b_1^1 b_1^2 \dots b_1^{n^2} v_2 e_2^1 e_2^2 e_2^3 v'_2 b_2^1 b_2^2 \dots b_2^{n^2} v_n e_n^1$   
 164  $e_n^2 e_n^3 v'_n)$ . Then, let  $T_{v_i}$  be the tree with leaves  $v_i$  and  $v'_i$  attached below the root,  $T_e$  be the tree  
 165 with leaves  $e_i^{i'}$  and  $e_j^{j'}$  attached below the root for each edge  $e = \{v_i, v_j\}$  of  $G$  (with  $i', j' \in$   
 166  $[1..3]$ ), and  $T_{B_i}$  be the tree with leaves  $b_i^1, \dots, b_i^{n^2}$  attached below the root for each  $i \in [1..n-1]$ .  
 167 We finally define  $T(G)$  as the tree such that  $T_{v_1}, T_{v_2}, \dots, T_{v_n}, T_{e_1}, T_{e_2}, \dots, T_{e_m}, T_{B_1}, T_{B_2}, \dots$   
 168 and  $T_{B_{n-1}}$  are attached below the root.

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

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

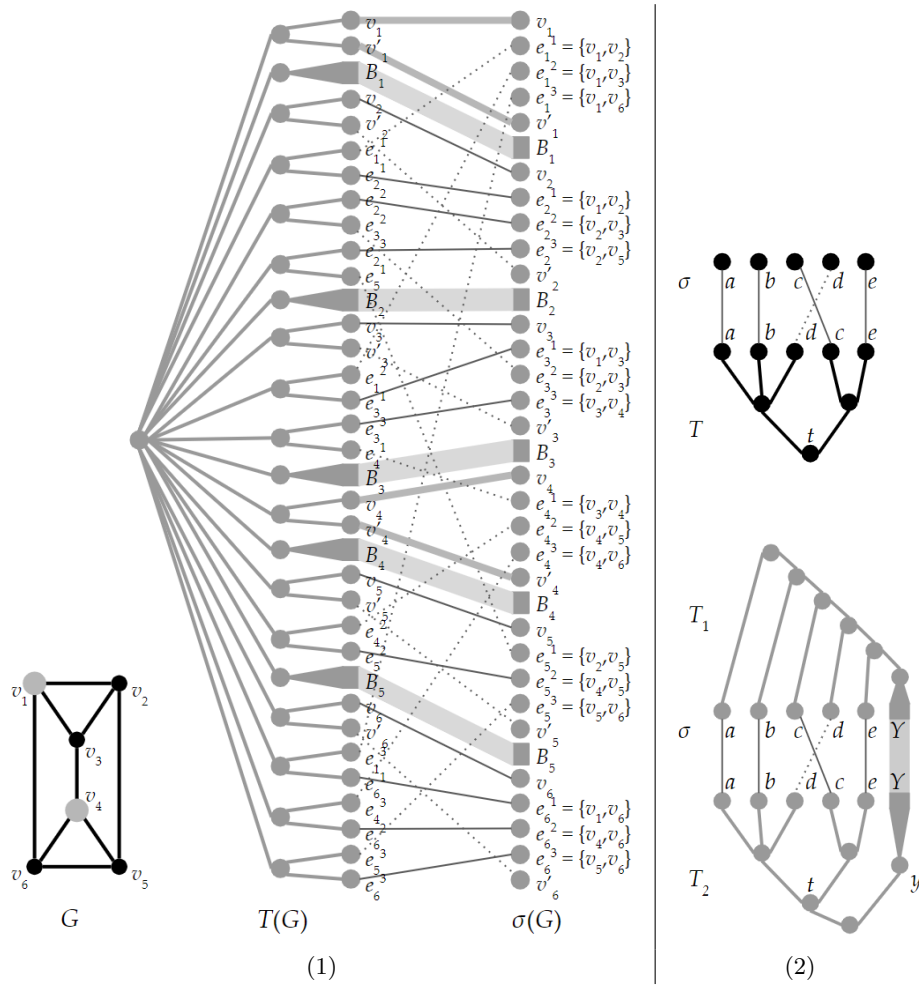
- 174 ■ for each edge  $e = \{v_i, v_j\} = e_i^{i'} = e_j^{j'}$  with  $i < j$ , we remove  $e_i^{i'}$  and call  $T_{e_j^{j'}} = T_e$  if  
 175  $v_i \in S$  or if neither  $v_i$  nor  $v_j$  belong to  $S$ ; and we remove  $e_j^{j'}$  and call  $T_{e_i^{i'}} = T_e$  if  $v_j \in S$   
 176 (as  $S$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

178 By ordering the children of the root of  $T(G)$  such as in Figure 2(1), that is by putting, for each  
 179  $v_i$  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  
 180 then  $T_{B_i}$  (except for  $i = n$ ), the order  $\sigma(G)$  restricted to the remaining  $m + n + k + n^2(n - 1)$   
 181 leaves is suitable on  $T'$ .

182  $\Leftarrow$ : Suppose that there exists a set  $L$  of at most  $m + n - k$  leaves such that  $\sigma(G)[X - L]$   
 183 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  
 184  $T$  such that  $\{p_{B_i}, v\}$  is a conflict wrt.  $\sigma(G)$ , we can delete this conflict either by deleting no  
 185 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  
 186 the set  $L$  of leaves to be deleted.

187 We now consider the trees  $T_{e_i}$  for each  $i \in [1..m]$ : by construction of  $\sigma(G)$ , as both leaves  
 188 of each such tree are separated by some  $B_{i'}$ , therefore by  $n^2 > m + n - k$  leaves, one of these  
 189 two leaves has to be removed, so it has to belong to  $L$ . We call  $L'$  the set of such leaves of  $L$ ,  
 190 therefore there exists a set  $L - L'$  of at most  $n - k$  other leaves to delete. So there exists a  
 191 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  
 192 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  
 193  $v_i$  and  $v'_i$  are not in  $L$  and all  $e_i^j$  are in  $L'$ . We claim that the vertices of  $G$  corresponding  
 194 to  $S_L$  are an independent set of  $G$ . Suppose for contradiction that it is not the case, then  
 195 there exists an edge  $e = e_i^{i'} = e_j^{j'}$  between two vertices  $v_i$  and  $v_j$  of  $G$ . By construction of  
 196  $L'$ , exactly one of the leaves labeled by  $e_i^{i'}$  and  $e_j^{j'}$  is in  $L'$  so the second one is in  $L - L'$ :  
 197 contradiction.  $\blacktriangleleft$

198  $\blacktriangleright$  **Corollary 2.** *The TTDE problem is NP-complete.*



■ **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  $(T, \sigma)$  (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  $\sigma$ , and  $T_2$  according to the input tree  $T$ .

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

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

208  $\Rightarrow$  Let  $X'$  be a size- $(n - k)$  subset of  $X$  such that  $\sigma[X']$  is suitable on  $T[X]$ . Then let  $\gamma$   
 209 be any order on  $Y$ : the concatenation  $\sigma[X']\gamma$  is suitable both on  $T_1[X' \cup Y]$  and  $T_2[X' \cup Y]$ ,

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

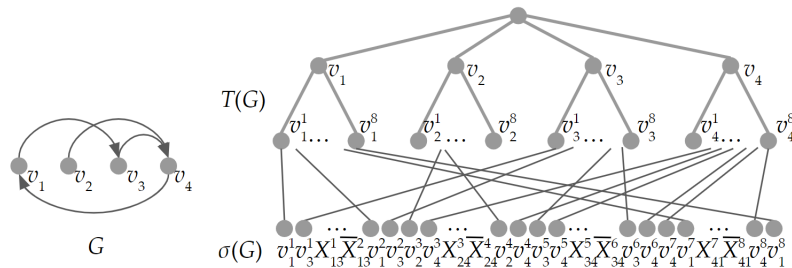
## 2.2 OTCM is NP-complete for trees with unbounded degree

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

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

232 We consider an instance of the FEEDBACK ARC SET problem, that is a directed graph  
 233  $G = (V = \{v_1, \dots, v_n\}, A)$  such that  $|A| = m$  and an integer  $f$ .

234 We now define an instance of the OTCM problem, illustrated in Figure 3. The set  $X$   
 235 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.  
 236 For each arc  $(v_i, v_j)$  of  $G$ , whose rank in the lexicographic order is  $k$ , we add to  $\sigma(G)$  a  $k^{\text{th}}$   
 237 supplementary ordered sequence (which we will later call a “block” corresponding to this arc)  
 238  $v_i^{2k-1} v_j^{2k-1} X_{i,j}^{2k-1} \overline{X}_{i,j}^{2k} v_i^{2k} v_j^{2k}$ , where  $X_{i,j}^{k'}$  is the ordered sequence of  $v_{i'}$  where  $i'$  ranges from  
 239 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  $v_{i'}$   
 240 where  $i'$  ranges from  $n$  down to 1, excluding  $i$  and  $j$ ). The tree  $T(G)$  is made of a root with  
 241  $n$  children  $v_1$  to  $v_n$ , each  $v_i$  having  $2m$  children, the leaves labeled by  $v_i^{k'}$  for  $k' \in [1..2m]$ .



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

242 Given an ordering  $\sigma'$  suitable for  $T$ , and an inversion  $(v_i^k, v_i^{k'})$  forming an inversion  
 243 between  $\sigma(G)$  and  $\sigma'$ , we say that this pair is *short-ranged* if  $k = k'$ , and *long-ranged*



244 otherwise. Furthermore, we say that  $\sigma'$  is *vertex-consistent* if, for every  $i$  and  $k < k'$ , we have  
 245  $\sigma'(v_i^k) < \sigma(v_i^{k'})$ . Finally, given  $\sigma'$ , we write  $\sigma''$  for the permutation of the  $[1..n]$  corresponding  
 246 to the children of the root.

247 We first claim that for any  $\sigma'$  suitable for  $T$ , there are at least  $2\binom{n}{2}\binom{2m}{2}$  long-range  
 248 inversions between  $\sigma'$  and  $\sigma(G)$ , and this bound is reached if  $\sigma'$  is vertex-consistent. Indeed,  
 249 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  
 250 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  
 251  $L(T, v_i)$  and  $L(T, v_{i'})$ , respectively). Overall, among  $4\binom{n}{2}\binom{2m}{2}$  such pairs of elements, there  
 252 are  $2\binom{n}{2}\binom{2m}{2}$  pairs creating an inversion (which is long-range by definition). For the case  
 253  $i = i'$ , note that pairs  $(v_i^k, v_i^{k'})$  do not create any inversion iff  $\sigma'$  is vertex-consistent, which  
 254 completes the proof of the claim.

255 Towards counting the number of short-ranged inversions, we say that an arc  $(v_i, v_j)$  of  
 256  $G$  is *satisfied* by  $\sigma''$  if  $\sigma''(i) < \sigma''(j)$ . Let  $i, j \in [1..n]$  and  $k \in [1..m]$ , and consider the two  
 257 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  
 258 same order in  $\sigma'$  (as defined by  $\sigma''$ ). If the  $k^{\text{th}}$  arc of  $G$  is  $(v_i, v_j)$ , then these two pairs  
 259 are also in the same order in  $\sigma$ , i.e., together they account for either 0 or 2 (short-ranged)  
 260 inversions. More precisely they yield 0 short-ranged inversions if  $(v_i, v_j)$  is satisfied by  
 261  $\sigma''$ , and 2 inversions otherwise. If the  $k^{\text{th}}$  arc of  $G$  is any other arc, then exactly one of  
 262  $(v_i^{2k-1}, v_j^{2k-1}), (v_i^{2k}, v_j^{2k})$  forms a short-ranged inversion. Overall a pair  $\{i, j\}$  such that one  
 263 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  
 264 one of  $(v_i, v_j), (v_j, v_i)$  is an unsatisfied arc yields  $m + 1$  short-range inversions, and any other  
 265 pair  $\{i, j\}$  with  $i \neq j$  yields  $m$  short-ranged inversions. Overall, if there are  $f$  unsatisfied  
 266 arcs,  $\sigma'$  yields  $\binom{n}{2}m - m + 2f$  inversions.

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

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

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

### 281 3 A polynomial-time algorithm for fixed-degree trees

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

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

289 **Proof.** Given a vertex  $v$  of a rooted tree  $T$ , a (strict or weak) order  $\sigma : L(T) \rightarrow [1..m]$  and  
 290 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  
 291 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  
 292  $\mathcal{X}(v, l, r)$  also depends on  $T$  and  $\sigma$  but we simplify the notation by not mentioning them as  
 293 they can clearly be identified from the context.

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

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

298 **Correctness:** We prove by induction on the size of  $L(v)$  that  $\mathcal{X}(v, l, r)$  is indeed a  
 299 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  
 300  $\forall \ell \in \mathcal{X}(v, l, r), \sigma(\ell) \in [l, r]$ .

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

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

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

#### 319 **4 An FPT algorithm for the deletion-degree parameter for OTDE**

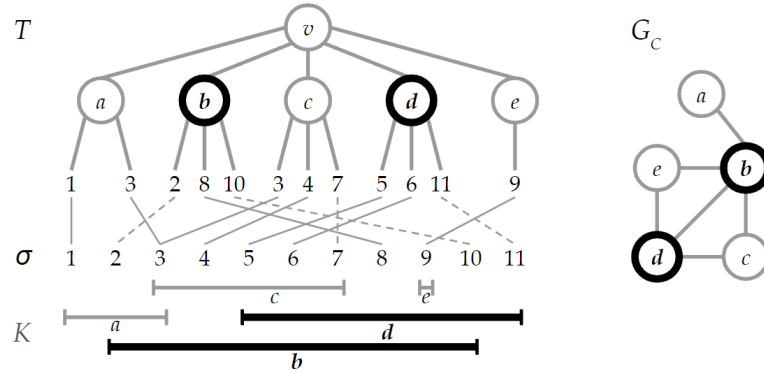
320 We recall that with a reduction of OTDE to 3-HITTING SET [10], using the best algorithm  
 321 known so far to solve this problem<sup>4</sup>, we can obtain an algorithm to solve OTDE  $O^*(2.08^k)$  [23],  
 322 where  $k$  is the number of leaves to delete and the  $O^*$  notation ignores the polynomial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

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

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

<sup>4</sup> <http://fpt.wikidot.com/fpt-races>

23:10 Reordering a tree according to an order on its leaves



■ **Figure 4** An instance  $(T, \sigma)$  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  $\sigma$  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.

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

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

340 **Proof.** Given a subset  $X'$  of  $X$ , we say that a node  $v$  of  $T$  has a *deletion* if some  $L(v) \not\subseteq X'$ ,  
 341 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  
 342 at least one of  $u, v$  has a deletion for  $X'$  (indeed, the conflict involves three leaves  $a, b, c$ , of  
 343 which at least one must be deleted). Thus, the vertices with a deletion in  $G_v$  form a vertex  
 344 cover of this graph. The lemma follows from the fact that at most  $\partial$  vertices have a deletion  
 345 in each conflict graph. ◀

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

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

354 We first compute  $\mathcal{X}(\{v\}, l, r)$  for each leaf  $v$ : clearly  $\mathcal{X}(\{v\}, l, r) = \{u\}$  if  $l \leq \sigma(v) \leq r$ ,  
 355 and  $\mathcal{X}(\{v\}, l, r) = \emptyset$  otherwise. For each internal vertex  $v$  (visiting the tree bottom-up), we

356 obtain  $\mathcal{X}(\{v\}, l, r)$  by first computing  $\mathcal{X}(P, l, r)$  for each prefix  $P$  of  $C_v$  by increasing order  
357 of size, using the following formulas:

$$\begin{aligned}
 358 \quad |\mathcal{X}(P, l, r)| &= \emptyset \text{ if } P = \emptyset \\
 359 \quad &= \max_{\substack{x \in [l, r], u \in P \\ P \setminus \{u\} \text{ prefix of } C_v}} |\mathcal{X}(P \setminus \{u\}, l, x)| + |\mathcal{X}(\{u\}, x, r)| \\
 360 \quad |\mathcal{X}(\{v\}, l, r)| &= |\mathcal{X}(C_v, l, r)| \\
 361
 \end{aligned}$$

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

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

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

- 373 (a)  $\text{span}_{X'}(c_i) \subseteq [x_{i-1}, x_i]$ , and  
374 (b)  $P_i = \{c_1, \dots, c_i\}$  is a prefix of the children of  $v$  wrt.  $\sigma$ .

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

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

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

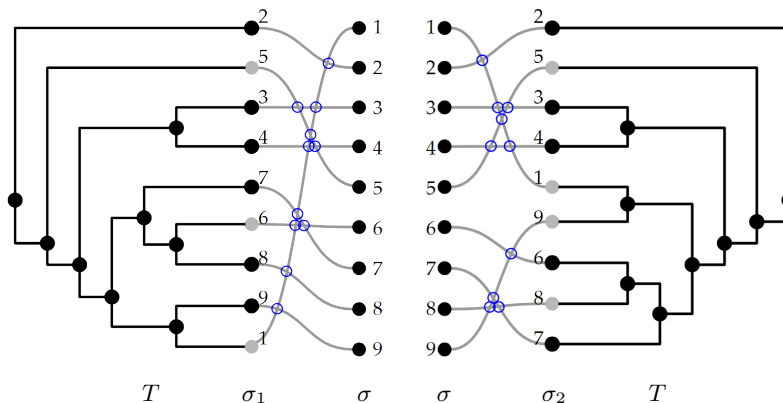
## 388 5 Optimizing OTCM and OTDE are two different things

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

396 We checked the optimality for both criteria by implementing the “naive” dynamic  
397 programming  $O(n^2)$  algorithm described in Section 2.1 of [10] to solve the OTCM problem  
398 and the  $O(n^4)$  algorithm described in Section 3 to solve the OTDE problem on binary trees.

## 23:12 Reordering a tree according to an order on its leaves

399 Both implementations are available in Python, under the GPLv3 licence, at [https://github.com/oseminck/tree\\_order\\_evaluation](https://github.com/oseminck/tree_order_evaluation), as well as the file `inputCounterExample1b.txt`  
 400 containing the Newick encoding for the tree of Figure 5.  
 401



■ **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  $\sigma$  suitable on  $T$  restricted to the remaining leaves, but the order  $\sigma_1$  suitable on  $T$  has 11 inversions, shown with empty circles, with  $\sigma$ ); the other one is optimal for the OTCM problem with the order  $\sigma_2$  suitable on  $T$  having 10 inversions with  $\sigma$  but not for the OTDE problem (4 leaves, for example the 4 gray ones, need to be deleted to make the order  $\sigma$  suitable on  $T$  restricted to the remaining leaves).

## 402 6 Experiments and discussion

403 In this section, we investigate the potential for use of OTCM and OTDE in applications  
 404 where the tree of elements is obtained from a clustering algorithm taking as input distances  
 405 between those elements, and where we want to test whether this clustering reflects some  
 406 intrinsic order on the elements, for example the chronological order. We both test the running  
 407 time of OTCM and OTDE on real data, and the performance of OTDE on simulated data  
 408 to detect possibly misplaced leaves in the order.

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

<sup>5</sup> 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.

tree	# leaves	OTCM time	# inversions	$p_{OTCM}$	OTDE time	# deleted leaves	$p_{OTDE}$
Séгур	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

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

420 instances of about 50 elements in the tree and in the order.

421 Investigating precisely whether the numbers of inversions or deleted leaves shown in  
 422 Table 1 are sufficiently small to reflect consistency with a chronological signal is beyond the  
 423 scope of this paper. However, we also provide  $p_{OTCM}$  and  $p_{OTDE}$ , the percentage of cases  
 424 when the best order on the leaves of the tree has the same number of inversions, or less  
 425 than the chronological order, among 10000 randomly generated orders for OTCM and 100  
 426 randomly generated orders for OTDE, respectively<sup>6</sup>. These numbers illustrate that in all  
 427 cases, it is unlikely that the observed optimal numbers of inversions or deleted leaves are due  
 428 to chance, as we get equal or smaller values of inversions or deleted leaves on less than 3% of  
 429 random orders (for Ponson du Terrail the number of inversions is 167 or less for 2.23% of  
 430 random orders; for one of the 10 000 simulated random orders, it reached as little as 124  
 431 inversions). These preliminary results obtained thanks to reasonably small running times  
 432 open new perspectives in investigating further the practical use of these algorithms, and  
 433 comparing their results with other methods to search for signals of chronological evolution in  
 434 textual data [21].

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

- 440 1. we randomly pick  $n$  distinct integers from the interval  $[0, 999]$ , which will be our set  $X$  of  
 441 leaves;
- 442 2. we build a distance matrix in which the distance between two elements from  $X$  is simply  
 443 the absolute difference between both; we add some noise to this matrix by adding or  
 444 subtracting in each cell a random quantity equal to at most 10% of the cell value, obtaining  
 445 a noisy matrix, from which we build an X-tree  $T$  using the `AgglomerativeClustering`  
 446 class from the package `sklearn`;
- 447 3. we randomly pick a set  $L_e$  of  $e$  leaves in  $X$  and replace their value by another integer,

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

## 23:14 Reordering a tree according to an order on its leaves

$n = \# \text{ leaves}$	$e = \# \text{ errors}$	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

■ **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).

448 randomly chosen from the interval  $[0, 999]$ , distinct from other leaf labels;  $\sigma$  is the set of  
 449 leaves ordered by increasing value taking into account these new values;

450 4. by solving the OTDE problem on  $T$  and  $\sigma$ , we compute the minimum set  $L$  of leaves to  
 451 remove to make  $\sigma[X - L]$  suitable on  $T[X - L]$ , and check whether  $L = L_e$ .

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

## 461 7 Conclusion and perspectives

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

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

476 ——— **References** ———

- 477 1 Mukul S Bansal, Wen-Chieh Chang, Oliver Eulenstein, and David Fernández-Baca. Generalized binary tanglegrams: Algorithms and applications. In *International Conference on*  
478 *Bioinformatics and Computational Biology*, pages 114–125. Springer, 2009. doi:10.1007/  
479 978-3-642-00727-9\_13.
- 481 2 Ziv Bar-Joseph, Erik D. Demaine, David K. Gifford, Nathan Srebro, Angèle M. Hamel, and  
482 Tommi S. Jaakkola. K-ary clustering with optimal leaf ordering for gene expression data.  
483 *Bioinformatics*, 19(9):1070–1078, 2003. doi:10.1093/bioinformatics/btg030.
- 484 3 Ziv Bar-Joseph, David K Gifford, and Tommi S Jaakkola. Fast optimal leaf ordering for hierar-  
485 chical clustering. *Bioinformatics*, 17(suppl 1):S22–S29, 2001. doi:10.1093/bioinformatics/  
486 17.suppl\_1.S22.
- 487 4 Ulrik Brandes. Optimal leaf ordering of complete binary trees. *Journal of Discrete Algorithms*,  
488 5(3):546–552, 2007. doi:10.1016/j.jda.2006.09.003.
- 489 5 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical*  
490 *Computer Science*, 411(40):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.
- 491 6 Tim Dwyer and Falk Schreiber. Optimal leaf ordering for two and a half dimensional phyloge-  
492 netic tree visualisation. In *APVis '04: Proceedings of the 2004 Australasian symposium on*  
493 *Information Visualisation*, volume 35, pages 109–115, 2004. doi:10.5555/1082101.1082114.
- 494 7 Denise Earle and Catherine B. Hurley. Advances in dendrogram seriation for application  
495 to visualization. *Journal of Computational and Graphical Statistics*, 24(1):1–25, 2015. doi:  
496 10.1080/10618600.2013.874295.
- 497 8 Maciej Eder, Jan Rybicki, and Mike Kestemont. Stylometry with R: a package for computa-  
498 tional text analysis. *R Journal*, 8(1):107–121, 2016. URL: [https://journal.r-project.org/  
499 archive/2016/RJ-2016-007/index.html](https://journal.r-project.org/archive/2016/RJ-2016-007/index.html).
- 500 9 Henning Fernau, Michael Kaufmann, and Mathias Poths. Comparing trees via crossing mini-  
501 mization. In *International Conference on Foundations of Software Technology and Theoretical*  
502 *Computer Science*, pages 457–469. Springer, 2005. doi:10.1007/11590156\_37.
- 503 10 Henning Fernau, Michael Kaufmann, and Mathias Poths. Comparing trees via crossing  
504 minimization. *Journal of Computer and System Sciences*, 76(7):593–608, 2010. doi:10.1016/  
505 j.jcss.2009.10.014.
- 506 11 Philippe Gambette, Olga Semince, Dominique Legallois, and Thierry Poibeau. Evaluat-  
507 ing hierarchical clustering methods for corpora with chronological order. In *EADH2021:*  
508 *Interdisciplinary Perspectives on Data. Second International Conference of the European*  
509 *Association for Digital Humanities*, Krasnoyarsk, Russia, September 2021. EADH. URL:  
510 <https://hal.archives-ouvertes.fr/hal-03341803>.
- 511 12 Michael Hahsler, Kurt Hornik, and Christian Buchta. Getting things in order: an introduction  
512 to the R package seriation. *Journal of Statistical Software*, 25(3):1–34, 2008. doi:10.18637/  
513 jss.v025.i03.
- 514 13 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer*  
515 *computations*, pages 85–103. Springer, 1972.
- 516 14 Cyril Labbé and Dominique Labbé. Existe-t-il un genre épistolaire? Hugo, Flaubert et  
517 Maupassant. In *Nouvelles Journées de l'ERLA*, pages 53–85. L'Harmattan, 2013.
- 518 15 Jean-Marc Leblanc. *Analyses lexicométriques des vœux présidentiels*. ISTE Group, 2016.
- 519 16 Bojan Mohar. Face covers and the genus problem for apex graphs. *Journal of Combinatorial*  
520 *Theory, Series B*, 82(1):102–117, 2001. doi:10.1006/jctb.2000.2026.
- 521 17 Hermann Moisl. How to visualize high-dimensional data: a roadmap. *Journal of Data Mining*  
522 *& Digital Humanities*, 2020. doi:10.46298/jdmh.5594.
- 523 18 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel,  
524 P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher,  
525 M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine*  
526 *Learning Research*, 12:2825–2830, 2011.



## 23:16 Reordering a tree according to an order on its leaves

- 527 **19** Ryo Sakai, Raf Winand, Toni Verbeiren, Andrew Vande Moere, and Jan Aerts. dendsort:  
528 modular leaf ordering methods for dendrogram representations in R. *F1000Research*, 3, 2014.  
529 doi:10.12688/f1000research.4784.1.
- 530 **20** Olga Seminck, Philippe Gambette, Dominique Legallois, and Thierry Poibeau. The corpus for  
531 idiolectal research (CIDRE). *Journal of Open Humanities Data*, 7:15, 2021. doi:10.5334/  
532 johd.42.
- 533 **21** Olga Seminck, Philippe Gambette, Dominique Legallois, and Thierry Poibeau. The evolution  
534 of the idiolect over the lifetime: A quantitative and qualitative study on French 19<sup>th</sup> century  
535 literature, 2022. Under review.
- 536 **22** Balaji Venkatachalam, Jim Apple, Katherine St John, and Dan Gusfield. Untangling tan-  
537 glegrams: comparing trees by their drawings. *IEEE/ACM Transactions on Computational*  
538 *Biology and Bioinformatics*, 7(4):588–597, 2010. doi:10.1109/TCBB.2010.57.
- 539 **23** Magnus Wahlström. *Algorithms, measures and upper bounds for satisfiability and related prob-*  
540 *lems*. PhD thesis, Department of Computer and Information Science, Linköpings universitet,  
541 2007. URL: <http://urn.kb.se/resolve?urn=urn%3Anbn%3Ase%3Aliu%3Adiva-8714>.