



**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. hal-03413413v1

**HAL Id: hal-03413413**

**<https://hal.science/hal-03413413v1>**

Submitted on 3 Nov 2021 (v1), last revised 15 Apr 2022 (v2)

**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.<sup>2</sup>

25 **2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact  
26 algorithms

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

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

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

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

---

<sup>1</sup> corresponding author

<sup>2</sup> The code is available in the following (anonymized) shared folder for evaluation purposes, the git repository will be made available in the final version. <https://www.dropbox.com/sh/6xkpiov9pbyrbf7/AABSjgQDSgGrvTgyDTvL6xdja?dl=0>

## 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 [6], under the name “one-layer STOP (stratified tree ordering problem)”. The authors provided an  $O(n^2)$  time 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 [21] 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 [3, 2, 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 directly the data used to build it, 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 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

81 OTDE problem parameterized by the deletion-degree of the solution, which is lower than  
 82 the number of leaves to delete, in Section 4. We also give an example of a tree and an order  
 83 built to have a distinct solution for the OTCM and OTDE problems in Section 5. Finally,  
 84 we illustrate the relevance of this problem, and of our implementations of algorithms solving  
 85 them, for applications in digital humanities, with experiments on trees built from literary  
 86 works, as well as simulated trees, in Section 6.

## 87 1.1 Definitions

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

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

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

102 Given two (strict or weak) orders  $\sigma_1$  and  $\sigma_2$  on  $X$  and two elements  $a \neq b$  of  $X$ , we say  
 103 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  $a <_{\sigma_2} b$ .

104 Given an  $X$ -tree  $T$ , a subset  $X'$  of  $X$  and an order  $\sigma$  on  $X$ , we denote by  $\sigma[X']$  the order  
 105  $\sigma$  restricted to  $X'$ , and by  $T[X']$  the tree  $T$  restricted to  $X'$ , that is the  $X'$ -tree obtained  
 106 from  $T$  by removing leaves labeled by  $X \setminus X'$  and contracting any arc to a non-labeled leaf,  
 107 from a degree-2 vertex or from the root if the root has degree 1. We define the *deletion-degree*  
 108 of  $X'$  as the maximum degree of the tree induced by the deleted leaves, i.e.  $T[X \setminus X']$ .  
 109 Intuitively, the deletion-degree measures how deletions in different branches converge on a  
 110 few nodes or if they merge progressively. Note that by definition, the deletion-degree of  $X'$   
 111 is upper-bounded both by the maximum degree of  $T$  and by the size of  $X \setminus X'$ .

112 We now define the two main problems addressed in this paper. As explained in the  
 113 introduction, we differ from previous definitions which considered two trees, one with a fixed  
 114 order on the leaves, as input, as only the leaf order of the second tree is useful to define the  
 115 problem and not the tree itself.

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

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

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

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

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

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

## 131 2 NP-hardness

### 132 2.1 OTDE and TTDE are NP-complete for trees with unbounded 133 degree

134 ► **Theorem 1.** *The OTDE problem is NP-complete for strict orders and a fortiori for weak  
135 orders.*

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

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

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

148 First, we define the strict order  $\sigma(G) = (v_1 e_1^1 e_1^2 e_1^3 v'_1 b_1^1 b_1^2 \dots b_1^{n^2} v_2 e_2^1 e_2^2 e_2^3 v'_2 b_2^1 b_2^2 \dots b_2^{n^2} v_n e_n^1$   
149  $e_n^2 e_n^3 v'_n)$ . Then, let  $T_{v_i}$  be the tree with leaves  $v_i$  and  $v'_i$  attached below the root,  $T_e$  be the tree  
150 with leaves  $e_i^{j'}$  and  $e_j^{j'}$  attached below the root for each edge  $e = \{v_i, v_j\}$  of  $G$  (with  $i', j' \in$   
151  $[1..3]$ ), and  $T_{B_i}$  be the tree with leaves  $b_i^1, \dots, b_i^{n^2}$  attached below the root for each  $i \in [1, n - 1]$ .  
152 We finally define  $T(G)$  as the tree such that  $T_{v_1}, T_{v_2}, \dots, T_{v_n}, T_{e_1}, T_{e_2}, \dots, T_{e_m}, T_{B_1}, T_{B_2}, \dots$   
153 and  $T_{B_{n-1}}$  are attached below the root.

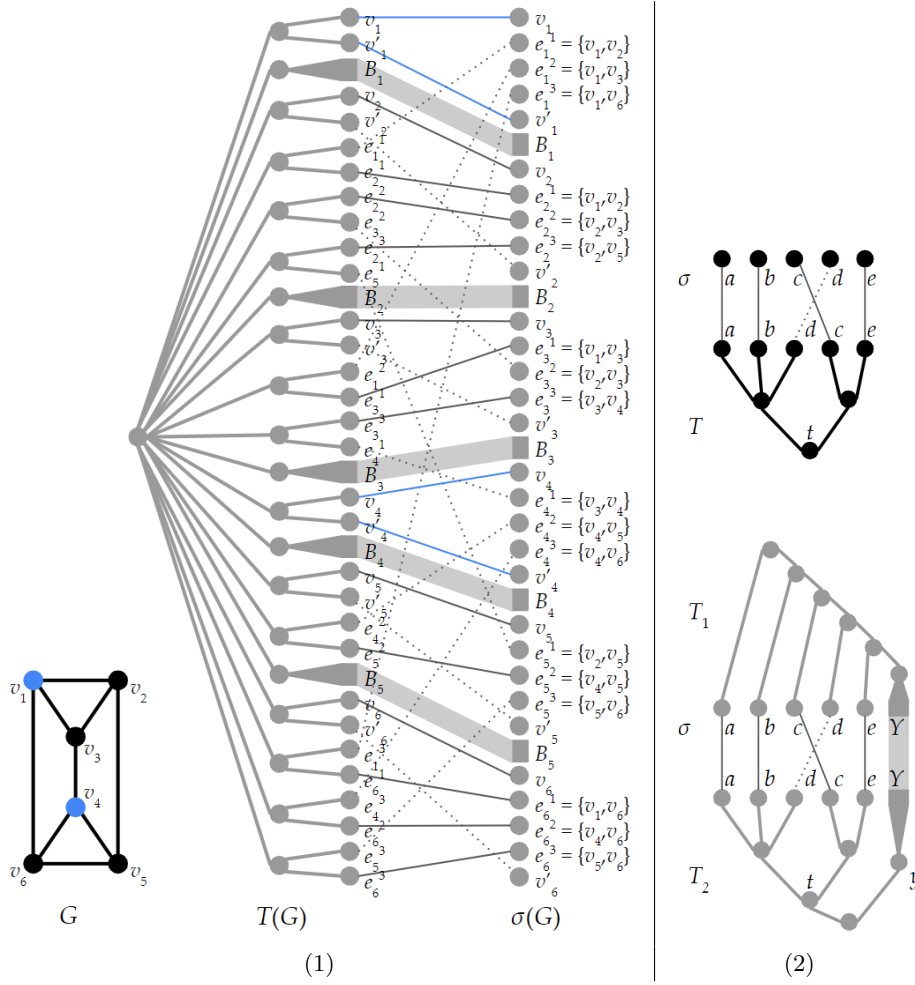
154 We claim that  $G$  has an independent set of size at least  $k \Leftrightarrow$  the instance  $(T(G), \sigma(G))$   
155 of the OTDE problem has a solution with a set  $L$  of at most  $m + n - k$  leaves to remove.

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

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

163 By ordering the children of the root of  $T(G)$  such as in Figure 1(1), that is by putting, for each  
164  $v_i$  with  $i \in [1, n]$ ,  $T_{v_i}$ , then  $T_{e_1^1}, T_{e_1^2}$  and  $T_{e_1^3}$  for each of the  $e_i^{i'}$  which were not removed and  
165 then  $T_{B_i}$  (except for  $i = n$ ), the order  $\sigma(G)$  restricted to the remaining  $m + n + k + n^2(n - 1)$   
166 leaves is suitable on  $T'$ .

167  $\Leftarrow$ : Suppose that there exists a set  $L$  of at most  $m + n - k$  leaves such that  $\sigma(G)[X - L]$   
168 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  
169  $T$  such that  $\{p_{B_i}, v\}$  is a conflict wrt.  $\sigma(G)$ , we can delete this conflict either by deleting no



■ **Figure 1** Illustration of the reductions of INDEPENDENT SET to OTDE and of OTDE to TTDE. (1) a graph  $G$  with independent set  $S = \{v_1, v_4\}$  of size 2 and the corresponding tree  $T(G)$  as well as the order  $\sigma(G)$  such that only the leaves connected with a dotted line to be deleted to make  $T(G)$  restricted to the remaining leaves suitable for the order. (2) Reduction from an OTDE instance  $(T, \sigma)$  to a TTDE instance  $(T_1, T_2)$ . 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  $T$ .

170 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  
 171 the set  $L$  of leaves to be deleted.

172 We now consider the trees  $T_{e_i}$  for each  $i \in [1..m]$ : by construction of  $\sigma(G)$ , as both leaves  
 173 of each such tree are separated by some  $B_{i'}$ , therefore by  $n^2 > m + n - k$  leaves, one of these  
 174 two leaves has to be removed, so it has to belong to  $L$ . We call  $L'$  the set of such leaves of  $L$ ,  
 175 therefore there exists a set  $L - L'$  of at most  $n - k$  other leaves to delete. So there exists a  
 176 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  
 177 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  
 178  $v_i$  and  $v'_i$  are not in  $L$  and all  $e_i^j$  are in  $L'$ . We claim that the vertices of  $G$  corresponding  
 179 to  $S_L$  are an independent set of  $G$ . Suppose by contradiction that it is not the case, then  
 180 there exists an edge  $e = e_i^{i'} = e_j^{j'}$  between two vertices  $v_i$  and  $v_j$  of  $G$ . By construction of  
 181  $L'$ , exactly one of the leaves labeled by  $e_i^{i'}$  and  $e_j^{j'}$  is in  $L'$  so the second one is in  $L - L'$ :

182 contradiction. ◀

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

184 **Proof.** TTDE is clearly in NP. We prove hardness by reduction from OTDE (see Figure 1(2)  
185 for an illustration). Consider an instance  $(T, \sigma)$  of OTDE with  $\sigma$  a strict order on  $n$  labels  
186  $X$ . Introduce a set  $Y$  of  $n$  new labels. Build  $T_1$  as a caterpillar with  $2n$  leaves (ordered from  
187 the leaf attached to the root to the leaf farthest from the root), where the first  $n$  leaves are  
188 labelled with  $X$  according to  $\sigma$  (in the same order), and the last  $n$  leaves are labelled with  
189  $Y$  (in any order). Build  $T_2$  as a tree, where the root has two children  $y, t$ , where  $y$  has  $n$   
190 children which are leaves labelled with  $Y$ , and  $t$  is the root of a subtree equal to  $T$ .

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

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

196  $\Leftarrow$  Let  $X', Y'$  be subsets of  $X, Y$  respectively and  $\sigma'$  be an order on  $X' \cup Y'$  such that  
197  $\sigma'$  is suitable on both  $T_1[X' \cup Y']$  and  $T_2[X' \cup Y']$ , and such that  $|X' \cup Y'| \geq 2n - k > n$   
198 (in particular,  $Y'$  contains at least one element denoted  $y$ , and  $|X'| \geq n - k$ ). From  $T_2$ , it  
199 follows that  $\sigma'$  is the concatenation (in any order) of an order  $\sigma_x$  of  $X'$  suitable for  $T[X']$   
200 and an order  $\sigma_y$  of  $Y'$ . Assume first that  $\sigma_x$  appears before  $\sigma_y$ . Then for each internal node  
201 of the caterpillar  $T_1$  with a child in  $X'$ , this child must be ordered before the other subtree  
202 (which contains  $y$ ). Thus, the nodes in  $X'$  are ordered according to  $\sigma[X']$ , and  $\sigma_x = \sigma[X']$ ,  
203 and  $T[X']$  is suitable with  $\sigma[X']$ . For the other case, where  $\sigma_y$  is ordered before  $\sigma_x$ , then for  
204 each node of the caterpillar with a child in  $X'$ , this child must be after the subtree containing  
205  $y$ , and the nodes in  $X'$  are ordered according to the reverse of  $\sigma[X']$  (i.e.  $\sigma_x = \overline{\sigma[X']}$ ). Thus,  
206 the reverse of  $\sigma[X']$  is suitable for  $T[X']$ , and  $\sigma[X']$  as well (this is obtained by reversing  
207 the permutation of all children of internal nodes of  $T$ ). In both cases,  $X'$  is a solution for  
208 OTDE( $T, \sigma$ ) with  $|X'| \geq n - k$ . ◀

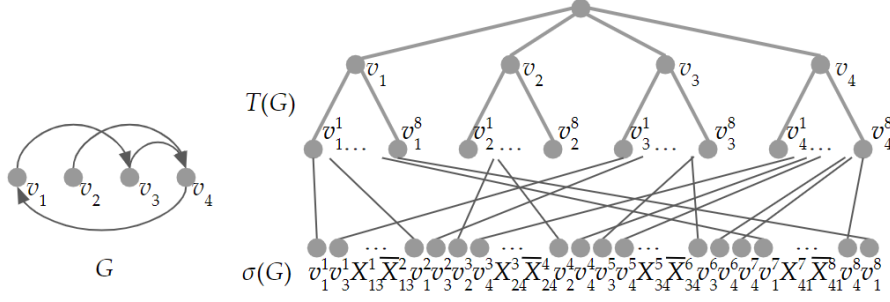
## 209 2.2 OTCM is NP-complete for trees with unbounded degree

210 ▶ **Theorem 3.** *The OTCM problem is NP-complete for strict orders and a fortiori for weak  
211 orders.*

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

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

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



**Figure 2** 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  $\sigma'$  suitable for  $T$ , and an inversion  $(v_i^k, v_{i'}^{k'})$  forming an inversion between  $\sigma(G)$  and  $\sigma'$ , we say that this pair is *short-ranged* if  $k = k'$ , and *long-ranged* otherwise. Furthermore, we say that  $\sigma'$  is *vertex-consistent* if, for every  $i$  and  $k < k'$ , we have  $\sigma'(v_i^k) < \sigma(v_{i'}^{k'})$ . Finally, given  $\sigma'$ , we write  $\sigma''$  for the permutation of the  $[1..n]$  corresponding to the children of the root.

We first claim that for any  $\sigma'$  suitable for  $T$ , there are at least  $2 \binom{n}{2} \binom{2m}{2}$  long-range inversions between  $\sigma'$  and  $\sigma(G)$ , and this bound is reached if  $\sigma'$  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 respectively in blocks  $k$  and  $k'$  of  $\sigma(G)$ ), and  $v_i^k <_{\sigma'} v_{i'}^{k'}$  iff  $\sigma''(i) < \sigma''(i')$  (since they are respectively in  $L(T, v_i)$  and  $L(T, v_{i'})$ ). 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  $\sigma'$  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  $\sigma''$  if  $\sigma''(i) < \sigma''(j)$ . Consider two pairs  $(v_i^{2k-1}, v_j^{2k-1})$  and  $(v_i^{2k}, v_j^{2k})$ . Then these two pairs are necessarily in the same order in  $\sigma'$ . If the  $k^{\text{th}}$  arc of  $G$  is  $(v_i, v_j)$ , then these two pairs are also in the same order in  $\sigma$ . Note that they (both) form an inversion iff  $(v_i, v_j)$  is not satisfied by  $\sigma''$ . If the  $k^{\text{th}}$  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 an 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-range 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  $m$  inversions. Overall, if there are  $f$  unsatisfied arcs,  $\sigma'$  yields  $\binom{m}{2} - m + 2f$  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  $\binom{n}{2} \binom{2m}{2} + \binom{m}{2} - m + 2f$  inversions.

$\Rightarrow$ : If  $G$  has a feedback arc set  $F$  of size  $f$ , as  $G[A - F]$  is acyclic, we consider an order  $\sigma''$  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 of the vertices in  $G[A - F]$ ). We now order the children  $v_i$  of the root of  $T(G)$  according to this order  $\sigma''$  and call  $\sigma'$  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  $\sigma'$  is vertex-consistent, and that an arc  $(v_i, v_j)$  is satisfied by  $\sigma''$  iff  $(v_i, v_j) \notin F$ . Thus,  $\sigma'$  yields  $\binom{n}{2} \binom{2m}{2} + \binom{m}{2} - m + 2f$  inversions.

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



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

We start by providing 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  $\sigma : L(T) \rightarrow [1..m]$  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  $\sigma$  but we simplify the notation by not mentioning them as they can clearly be identified from the context.

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

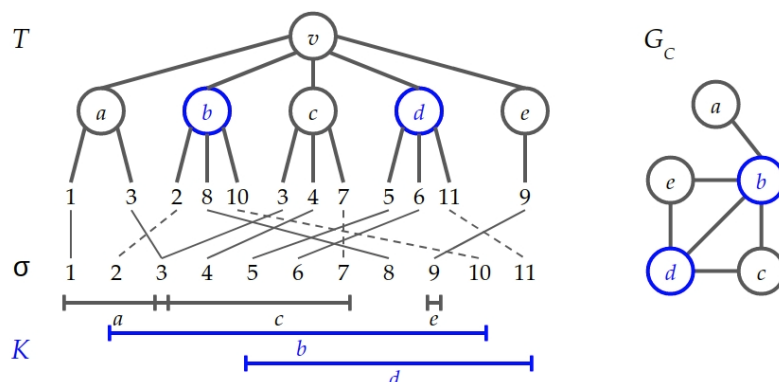
$$\begin{aligned} \blacksquare \quad |\mathcal{X}(v, l, r)| &= \max_{\substack{\text{permutation } \pi \text{ of } [1..k] \\ x_1=l \leq x_2 \leq \dots \leq x_k \leq x_{k+1}=r}} \sum_{i=1}^k |\mathcal{X}(c_{\pi(i)}, x_i, x_{i+1})| \text{ if } v \text{ is an internal node of } T; \\ \blacksquare \quad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

**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 by contradiction that there exists a set of integers  $l \leq r$  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  $X'_1, \dots, X'_k$  the sets of leaves  $L(c_1) \cap X', \dots, L(c_k) \cap X'$  respectively. Without loss of generality we consider that the children  $c_i$  of  $v$  are labeled such that  $\max_{\ell \in X'_i} \{\sigma(\ell)\} \leq \min_{\ell \in X'_{i+1}} \{\sigma(\ell)\}$ . 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  $i \in [1..k]$ ,  $|X'_i| \leq |\mathcal{X}(v, \min_{\ell \in X'_i} \{\sigma(\ell)\}, \max_{\ell \in X'_i} \{\sigma(\ell)\})|$ , so  $|X'_i| \leq |\mathcal{X}(v, m_i, m_{i+1})|$  because  $[\min_{\ell \in X'_i} \{\sigma(\ell)\}, \max_{\ell \in X'_i} \{\sigma(\ell)\}] \subseteq [m_i, m_{i+1}]$ . Therefore,  $|X'| = \sum_{i=1}^k |X'_i| \leq \sum_{i=1}^k |\mathcal{X}(v, m_i, m_{i+1})|$  so by definition of  $\sigma[\mathcal{X}(v, l, r)]$ ,  $|X'| \leq \sigma[\mathcal{X}(v, l, r)]$ : contradiction!

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  $O(d!n^{d+2})$ . ◀



**Figure 3** An instance  $(T, \sigma)$  of OTDE (top-left), with a vertex  $v$  having children set  $C = \{a, b, c, d, e\}$ . The conflict graph  $G_C$  of  $C$  (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  $C$  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  $C$ . 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.

#### 299 **4** An FPT algorithm for the *deletion-degree* parameter for OTDE

300 We recall that with a reduction of OTDE to 3-HITTING SET [10], using the best algorithm  
 301 known so far to solve this problem<sup>3</sup>, we can obtain an  $O^*(2.08^k)$  [22] algorithm to solve  
 302 OTDE, where  $k$  is the number of leaves to delete. In this section we obtain an FPT algorithm  
 303 in time  $O(n^4 d \partial^{2\partial})$ , where  $d$  is the maximum degree of the tree and  $\partial$  is the deletion-degree  
 304 of the solution.

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

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

309 We first introduce some definitions (see Figure 3 for a illustration of these definitions  
 310 and the algorithm in general). Given any vertex  $v$  of  $T$ , let  $C_v$  be the (independent) set of  
 311 children of  $v$ , and let  $G_v$  be the *conflict graph* with vertex set  $C_v$  and with one edge per  
 312 conflict. Let  $K$  be a vertex cover of  $C_v$ . Then the vertices of  $C_v \setminus K$  have a *canonical order*  
 313  $(w_1, \dots, w_{k'})$ , with  $k' = |C_v \setminus K|$  and  $w_i \preceq_\sigma w_j$  for all  $i \leq j$  (ties may happen when two  
 314 children contain a single leaf each which are equal, such ties are broken arbitrarily). We say  
 315 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  $i \leq k'$ ,  
 316  $P \setminus K = \{w_1, \dots, w_i\}$ ). In other words, ignoring all subtrees below vertices of  $K$ , all leaves  
 317 below vertices of a prefix  $P$  are necessarily ordered before leaves below vertices outside of  $P$ .

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

320 **Proof.** Given a subset  $X'$  of  $X$ , we say that a node  $v$  of  $T$  has a *deletion* if some  $L(v) \not\subseteq X'$ ,  
 321 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

<sup>3</sup> <http://fpt.wikidot.com/fpt-races>

322 at least one of  $u, v$  has a deletion for  $X'$  (indeed, the conflict involves three leaves  $a, b, c$ , of  
 323 which at least one must be deleted). Thus, the vertices with a deletion in  $G_v$  form a vertex  
 324 cover of this graph. The lemma follows from the fact that at most  $\partial$  vertices have a deletion  
 325 in each conflict graph. ◀

326 The first step of our algorithm consists in computing, for each node  $v$  of the graph, the  
 327 set  $C$  of children of  $v$ , its conflict graph  $G_v$ , and a minimum vertex cover  $K_v$  of  $G_C$ . Since  
 328 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],  
 329 and overall this first step takes  $O(1.3^\partial n + \partial n^2)$ .

330 We proceed with the dynamic programming part of our algorithm. To this end, we  
 331 generalize the table  $\mathcal{X}$  to sets of nodes (instead of only  $v$ ) as follows:  $\mathcal{X}(P, l, r)$  corresponds  
 332 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  
 333 a node  $v$  with children set  $C$ ,  $\mathcal{X}(v, l, r) = \mathcal{X}(\{v\}, l, r) = \mathcal{X}(C, l, r)$ .

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

$$\begin{aligned}
 338 \quad |\mathcal{X}(P, l, r)| &= \emptyset \text{ if } P = \emptyset \\
 339 \quad &= \max_{\substack{x \in [l..r], u \in P \\ P \setminus \{u\} \text{ prefix of } C_v}} |\mathcal{X}(P \setminus \{u\}, l, x)| + |\mathcal{X}(\{u\}, x, r)| \\
 340 \quad |\mathcal{X}(\{v\}, l, r)| &= |\mathcal{X}(C_v, l, r)| \\
 341
 \end{aligned}$$

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

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

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

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

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

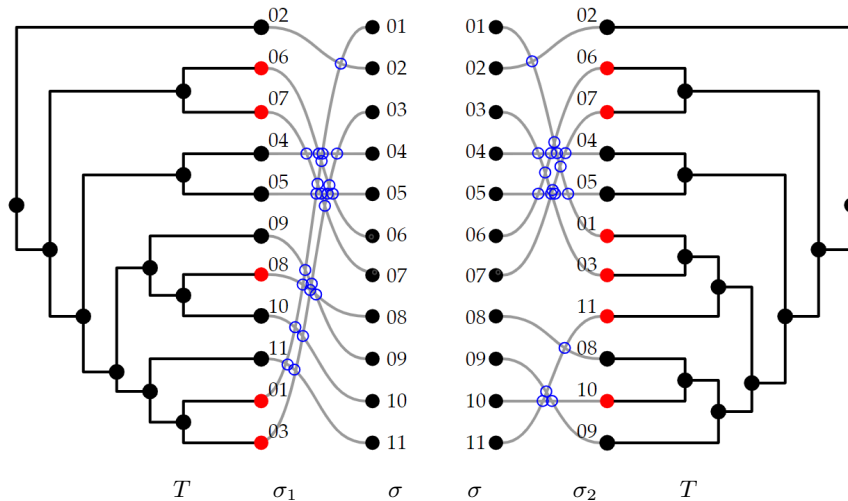
362 Condition (a) follows from the fact that  $X'$  is a solution for  $OTDE(T, \sigma)$ , so that the  
 363 span covered by the leaves under siblings do not overlap. For condition (b) we refer to the  
 364 definition of prefix: each  $P_i \setminus K_c$  is indeed a prefix in the canonical ordering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

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

## 368 5 Optimizing OTCM and OTDE are two different things

369 In order to ensure that OTCM and OTDE are actually optimizing different criteria, we  
 370 provide in Figure 4 an example of  $X$ -tree and an order of its leaves where the order reaching  
 371 the best  $k$  for a positive answer of the OTCM problem does not provide the optimal value  
 372 for the number of leaves to delete in a positive answer of OTDE and where the best  $k$  for a  
 373 positive answer of the OTDE problem does not provide an optimal value for the number of  
 374 inversions for a positive answer of the OTCM problem.

375 We checked the optimality for both criteria by implementing the “naive” dynamic  
 376 programming  $O(n^2)$  algorithm described in Section 2.1 of [10] to solve the OTCM problem  
 377 and the  $O(n^4)$  algorithm described in Section 3 to solve the OTDE problem on binary trees.  
 378 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 `inputCounterExample1.txt`  
 379 containing the Newick encoding for the tree of Figure 4.  
 380



381 **Figure 4** Two planar embeddings of a rooted tree  $T$ : the one on the left is optimal for the OTDE  
 382 problem (deleting the 5 red leaves makes the order  $\sigma$  suitable on  $T$  restricted to the remaining leaves,  
 383 but the order  $\sigma_1$  suitable on  $T$  has 22 inversions, shown with blue circles, with  $\sigma$ ); the other one is  
 384 optimal for the OTCM problem with the order  $\sigma_2$  suitable on  $T$  having 17 inversions with  $\sigma$  but not  
 385 for the OTDE problem (6 leaves, for example the 6 red ones, need to be deleted to make the order  $\sigma$   
 386 suitable on  $T$  restricted to the remaining leaves).

## 381 6 Experiments and discussion

382 In this section, we investigate the potential for use of OTCM and OTDE in applications  
 383 where clustering algorithms are used on distance data which is supposed to reflect some  
 384 intrinsic order on the elements, for example the chronological order. We both test the running  
 385 time of OTCM and OTDE on real data, and the performance of OTDE on simulated data  
 386 to detect possibly misplaced leaves in the order.

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

■ **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.

387 The first experiment deals with text data: the CIDRE corpus [20] that contains the works  
388 of 11 French 19<sup>th</sup> century fiction writers dated by year (every file contains a book that is  
389 annotated with its year of writing). We apply apply hierarchical clustering on the different  
390 corpora using the `AgglomerativeClustering` class from the package `sklearn` [18]. Distance  
391 matrices on which the clustering is based are obtained by using the relative frequencies  
392 of the 500 most frequent tokens<sup>4</sup> in each corpus. Distance matrices were generated using  
393 the R package `stylo` [8], with the *canberra* distance metric. We obtain the results given in  
394 Table 1, which provide the running time in milliseconds of the algorithms we implemented  
395 to solve OTCM and OTDE. They show that both algorithms on binary trees are quick  
396 enough to handle typical instances of the OTCM and the OTDE problems relevant for digital  
397 humanities, a few milliseconds for the first one and a few seconds for the second one, for  
398 instances of about 50 elements in the tree and in the order.

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

- 404 1. we randomly pick  $n$  distinct integers from the interval  $[0, 999]$ , which will be our set  $X$  of  
405 leaves;
- 406 2. we build a distance matrix in which the distance between two elements from  $X$  is simply  
407 the absolute difference between both; we add some noise to this matrix by adding or  
408 subtracting in each cell a random quantity equal to at most 10% of the cell value, obtaining  
409 a noisy matrix, from which we build an X-tree  $T$  using the `AgglomerativeClustering`  
410 class from the package `sklearn`;
- 411 3. we randomly pick a set  $L_e$  of  $e$  leaves in  $X$  and replace their value by another integer,  
412 randomly chosen from the interval  $[0, 999]$ , distinct from other leaf labels;  $\sigma$  is the set of  
413 leaves ordered by increasing value taking into account these new values;

<sup>4</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", "text", "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.

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

414 4. by solving the OTDE problem on  $T$  and  $\sigma$ , we compute the minimum set  $L$  of leaves to  
 415 remove to make  $\sigma[X - L]$  suitable on  $T[X - L]$ , and check whether  $L = L_e$ .  
 416 This experiment simulates the situation where we would have dating errors on the elements  
 417 we clustered in a tree. Note that like in the case of dating errors, the error in our simulation  
 418 may not change the overall order on the leaves. Table 2 provides, for each chosen values of  
 419  $n$  and  $e$ , the proportion of simulated instances of *OTDE* where  $L = L_e$ , that is when our  
 420 algorithm removed exactly the  $e$  leaves whose label had been randomly modified. We can  
 421 observe that this happens in a majority of cases only when the number of modified leaves is  
 422 small compared with the total number of leaves (up to 2 for 20 leaves, up to 4 for 50 leaves).  
 423 Solving OTDE still allows to identify  $e - 1$  among the  $e$  modified leaves in a majority of  
 424 cases in all our experiments.

## 425 7 Conclusion and perspectives

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

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

## 440 References

- 441 1 Mukul S Bansal, Wen-Chieh Chang, Oliver Eulenstein, and David Fernández-Baca. Gen-  
 442 eralized binary tanglegrams: Algorithms and applications. In *International Conference on*

- 443 *Bioinformatics and Computational Biology*, pages 114–125. Springer, 2009. doi:10.1007/  
444 978-3-642-00727-9\_13.
- 445 2 Ziv Bar-Joseph, Erik D. Demaine, David K. Gifford, Nathan Srebro, Angèle M. Hamel, and  
446 Tommi S. Jaakkola. K-ary clustering with optimal leaf ordering for gene expression data.  
447 *Bioinformatics*, 19(9):1070–1078, 2003. doi:10.1093/bioinformatics/btg030.
- 448 3 Ziv Bar-Joseph, David K Gifford, and Tommi S Jaakkola. Fast optimal leaf ordering for hierarchi-  
449 cal clustering. *Bioinformatics*, 17(suppl\_1):S22–S29, 2001. doi:10.1093/bioinformatics/  
450 17.suppl\_1.S22.
- 451 4 Ulrik Brandes. Optimal leaf ordering of complete binary trees. *Journal of Discrete Algorithms*,  
452 5(3):546–552, 2007. doi:10.1016/j.jda.2006.09.003.
- 453 5 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical*  
454 *Computer Science*, 411(40):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.
- 455 6 Tim Dwyer and Falk Schreiber. Optimal leaf ordering for two and a half dimensional phylo-  
456 genetic tree visualisation. In *APVis '04: Proceedings of the 2004 Australasian symposium on*  
457 *Information Visualisation*, volume 35, pages 109–115, 2004. doi:10.5555/1082101.1082114.
- 458 7 Denise Earle and Catherine B. Hurley. Advances in dendrogram seriation for application  
459 to visualization. *Journal of Computational and Graphical Statistics*, 24(1):1–25, 2015. doi:  
460 10.1080/10618600.2013.874295.
- 461 8 Maciej Eder, Jan Rybicki, and Mike Kestemont. Stylometry with r: a package for computa-  
462 tional text analysis. *R Journal*, 8(1):107–121, 2016. URL: [https://journal.r-project.org/  
463 archive/2016/RJ-2016-007/index.html](https://journal.r-project.org/archive/2016/RJ-2016-007/index.html).
- 464 9 Henning Fernau, Michael Kaufmann, and Mathias Poths. Comparing trees via crossing minim-  
465 ization. In *International Conference on Foundations of Software Technology and Theoretical*  
466 *Computer Science*, pages 457–469. Springer, 2005. doi:10.1007/11590156\_37.
- 467 10 Henning Fernau, Michael Kaufmann, and Mathias Poths. Comparing trees via crossing  
468 minimization. *Journal of Computer and System Sciences*, 76(7):593–608, 2010. doi:10.1016/  
469 j.jcss.2009.10.014.
- 470 11 Philippe Gambette, Olga Seminck, Dominique Legallois, and Thierry Poibeau. Evaluat-  
471 ing hierarchical clustering methods for corpora with chronological order. In *EADH2021:*  
472 *Interdisciplinary Perspectives on Data. Second International Conference of the European*  
473 *Association for Digital Humanities*, Krasnoyarsk, Russia, September 2021. EADH. URL:  
474 <https://hal.archives-ouvertes.fr/hal-03341803>.
- 475 12 Michael Hahsler, Kurt Hornik, and Christian Buchta. Getting things in order: an introduction  
476 to the R package seriation. *Journal of Statistical Software*, 25(3):1–34, 2008. doi:10.18637/  
477 jss.v025.i03.
- 478 13 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer*  
479 *computations*, pages 85–103. Springer, 1972.
- 480 14 Cyril Labbé and Dominique Labbé. Existe-t-il un genre épistolaire? Hugo, Flaubert et  
481 Maupassant. In *Nouvelles Journées de l'ERLA*, pages 53–85. L'Harmattan, 2013.
- 482 15 Jean-Marc Leblanc. *Analyses lexicométriques des vœux présidentiels*. ISTE Group, 2016.
- 483 16 Bojan Mohar. Face covers and the genus problem for apex graphs. *Journal of Combinatorial*  
484 *Theory, Series B*, 82(1):102–117, 2001. doi:10.1006/jctb.2000.2026.
- 485 17 Hermann Moisl. How to visualize high-dimensional data: a roadmap. *Journal of Data Mining*  
486 *& Digital Humanities*, 2020. doi:10.46298/jdmh.5594.
- 487 18 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel,  
488 P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher,  
489 M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine*  
490 *Learning Research*, 12:2825–2830, 2011.
- 491 19 Ryo Sakai, Raf Winand, Toni Verbeiren, Andrew Vande Moere, and Jan Aerts. dendsort:  
492 modular leaf ordering methods for dendrogram representations in R. *F1000Research*, 3, 2014.  
493 doi:10.12688/f1000research.4784.1.

- 494 20 Olga Seminck, Philippe Gambette, Dominique Legallois, and Thierry Poibeau. The corpus for  
495 idiolectal research (CIDRE). *Journal of Open Humanities Data*, 7:15, 2021. doi:10.5334/  
496 johd.42.
- 497 21 Balaji Venkatachalam, Jim Apple, Katherine St John, and Dan Gusfield. Untangling tangle-  
498 grams: comparing trees by their drawings. *IEEE/ACM Transactions on Computational Biology  
499 and Bioinformatics*, 7(4):588–597, 2010. doi:10.1109/TCBB.2010.57.
- 500 22 Magnus Wahlström. *Algorithms, measures and upper bounds for satisfiability and related prob-  
501 lems*. PhD thesis, Department of Computer and Information Science, Linköpings universitet,  
502 2007. URL: <http://urn.kb.se/resolve?urn=urn%3Anbn%3Ase%3Aliu%3Adiva-8714>.