



HAL
open science

An FPT Algorithm for Spanning, Steiner and Other subTree Problems Parameterized with the Treewidth.

Dimitri Watel

► **To cite this version:**

Dimitri Watel. An FPT Algorithm for Spanning, Steiner and Other subTree Problems Parameterized with the Treewidth.. 2020. hal-02610732

HAL Id: hal-02610732

<https://hal.science/hal-02610732>

Preprint submitted on 17 May 2020

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 An FPT Algorithm for Spanning, Steiner and 2 Other subTree Problems Parameterized with the 3 Treewidth.

4 **Dimitri Watel**

5 ENSIE

6 SAMOVAR

7 dimitri.watel@ensiie.fr

8 — Abstract —

9 This paper investigates the possibility to find a single FPT algorithm with respect to the treewidth
10 that solves a large variety of spanning tree, steiner tree and more generally covering tree problems
11 that can be found in the literature. This includes problems for which no such algorithm was already
12 described as the Minimum Branch Vertices problem, the Minimum Leaf Spanning Tree problem or
13 the k -Bottleneck Steiner Tree Problem. To do so, a generalization of many of those covering tree
14 problems, called the Minimum subTree problem with Degree Weights MTDW, is introduced and the
15 parameterized complexity of that problem is studied.

16 **2012 ACM Subject Classification** Theory of computation → Complexity classes; Theory of com-
17 putation → Parameterized complexity and exact algorithms; Theory of computation → Graph
18 algorithms analysis; Theory of computation → Dynamic programming

19 **Keywords and phrases** Parameterized complexity, Treewidth, Spanning tree, Dynamic programming

20 **Digital Object Identifier** 10.4230/LIPIcs...

21 **1** Introduction

22 There exists a real variety of spanning tree, steiner tree and more generally covering tree
23 problems that can be found in the literature and mostly have applications in network routing.
24 In each such problem, the objective is to find a subtree in a graph satisfying some constraints
25 and minimizing an objective. Well known examples are the Minimum Undirected Steiner
26 Tree problem (UST) in which we search for a minimum-edge-cost subtree of an undirected
27 graph covering a specific subset of nodes; the k -Minimum Spanning Tree problem (k -MST)
28 in which we search again for a minimum-edge-cost subtree covering any k nodes; the Prize
29 Collecting Steiner Tree problem (PCST) in which the edges and node are weighted, and
30 adding an edge to the tree costs the weight of that edge, but not covering a node costs the
31 weight of that node; the Minimum Branch Vertices problem (MBV) in which the tree must
32 span all the nodes and minimize the number of nodes with degree 3 or more; or the Minimum
33 Leaf Spanning Tree (MLST) in which we minimize the number of leaves.

34 A natural question to ask is how hard are those problems and their variants when the
35 graph is close to a tree. A way to describe the distance between a graph and a tree is the
36 treewidth, introduced by Robertson and Seymour [8], and actively used in parameterized
37 complexity of graph optimization problems [3, 4]. It was proved, for instance, that UST,
38 PCST and k -MST are FPT with respect to the treewidth [2, 7]. No such result seems to exist
39 for MBV or MLST. However, the last two problems are a generalization of the Hamiltonian
40 path problem which is also FPT in the treewidth [4]. This paper aims to explore the fact
41 that all those problems can be described (or rewritten) only by looking at the degree of the
42 nodes of the graph in the tree. As shown in the following sections, that common property
43 makes all those problems, and most of their variants, FPT with respect to the treewidth.



© Dimitri Watel;

licensed under Creative Commons License CC-BY

Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

44 **Contributions of the paper**

45 We introduce the Minimum subTree problem with Degree Weights (MTDW). This problem
 46 encodes many kinds of constraints (for instance spanning, degree or cost constraints), that
 47 must be satisfied by a feasible tree, by associating to each node a set of scores depending on
 48 the degree of the node in the tree. We then get a set of scores of the tree by summing the
 49 scores of the nodes. One of the scores is used to define an objective function that must be
 50 minimized, and the others are used to define a set of constraint.

51 Given an undirected graph G and a node of v , we denote by $d_G(v)$ and $\gamma_G(v)$ the degree
 52 and the incident edges of v in G . We are given an undirected graph $G = (V, E)$ with n nodes,
 53 an integer $m \geq 0$, $m + 1$ mappings $C_1, C_2, \dots, C_m, C_{m+1}$ associating to each node $v \in V$ and
 54 to each integer $d \in \llbracket 0; d_G(v) \rrbracket$ an integer $C_j(v, d) \in \mathbb{Z}$, and m integers $K_1, K_2, \dots, K_m \in \mathbb{Z}$.
 55 We search for a tree T included in G such that, for $j \in \llbracket 1; m \rrbracket$, $\sum_{v \in V} C_j(v, d_T(v)) \leq K_j$, and
 56 minimizing $\sum_{v \in V} C_{m+1}(v, d_T(v))$. Note all the nodes of the graph intervene in the formulas,
 57 including those for which $d_T(v) = 0$.

58 For instance the Minimum Leaf Spanning Tree problem can be rewritten as a subproblem
 59 of MTDW with $m = 1$. C_1 is a spanning tree constraint: $C_1(v, 0) = 1$, $C_1(v, d \geq 1) = 0$ and
 60 $K_1 = 0$. We minimize the number of leaves with C_2 : $C_2(v, 1) = 1$ and $C_2(v, d \neq 1) = 0$.

61 In this paper, we mostly focus on the parameterized complexity of MTDW with respect
 62 to the treewidth and proves that a large set of subproblems of MTDW are FPT when
 63 parameterized with the treewidth, including all of the previously mentioned problems. More
 64 precisely, three parameters are studied: the treewidth TW of G , the number of constraints m
 65 and the maximum degree Δ above which every mapping C_j is constant: for every $j \in \llbracket 1; m \rrbracket$,
 66 $v \in V$ and $d \geq \Delta$, $C_j(v, d) = C_j(v, \Delta)$. Throughout the paper, we distinguish three possible
 67 cases for a parameter of MTDW depending if we are restricted to the instances where
 68 that parameter equals a constant, in which case we write the parameter on the left (for
 69 instance $(\Delta = 2)$ -MTDW) or if the parameter is classically considered from a parameterized
 70 complexity point of view, in which case, we explicitly mention it as a parameter. A last
 71 element that affects the complexity results in this paper is the encoding of the values K_j
 72 and $C_j(v, d)$ for every $j \in \llbracket 1; m \rrbracket$, $v \in G$ and $d \leq d_G(v)$. Some hardness results do not hold
 73 if those values are unary encoded. Let $\max |C| = \max_{j=1}^m \sum_{v \in V} \sum_{d \leq d_G(v)} |C_j(d, v)|$. Every
 74 result explicitly specifies if $\max |C|$ is unary or binary, meaning that every mapping C_j is
 75 unary or binary encoded. We may assume, without loss of generality, that $|K_j| \leq \max |C|$
 76 for every $j \in \llbracket 1; m \rrbracket$, as, otherwise, either the j -th constraint is necessarily satisfied or
 77 necessarily unsatisfied, thus the encoding of those integers is never given. Note also that the
 78 mapping C_{m+1} is not included in the formula of $\max |C|$: the cost function is always binary.

79 The next section provides the following theorem.

80 **► Theorem 1.** *If $\max |C|$ is unary, MTDW is XP with respect to TW and m , and, for every*
 81 *$c \in \mathbb{N}$, $(m = c)$ -MTDW is FPT with respect to TW and Δ .*

82 This theorem can be applied to all the previously cited problems as they can be rewritten
 83 as subproblem of MTDW with a fixed value of m . Appendix A details, for each mentioned
 84 subproblem, the consequences of this Theorem. In short, it gives, in addition to all the
 85 existing results, an FPT algorithm with respect to the treewidth to solve a large class of
 86 subtree problems. The last section of the paper gives hardness results, proving that it is
 87 not possible to change the encoding of $\max |C|$, or to consider that Δ or TW is part of
 88 the instance and keep MTDW in the class FPT: the problem is either NP-Hard or XP but
 89 $W[1]$ -hard with respect to the parameters.

2 An FPT Algorithm for $(m = c)$ -MTDW with Respect to Δ and TW

In this part, we provide an algorithm that proves Theorem 1.

Let $\mathcal{I} = (G = (V, E), C_1, C_2, \dots, C_{m+1}, K_1, K_2, \dots, K_m)$ be an instance of MTDW. Let τ be a *tree composition* of G . We will solve \mathcal{I} by using a dynamic programming algorithm on a tree decomposition of the graph. In order to avoid any confusion, a node of τ will be called a *bag*. We recall that τ is a tree, that every node belongs to at least one bag, that for each edge $(v, w) \in E$, there exists a bag of τ containing v and w , and that the subgraph of τ with all the bags containing a same node v is connected. For each bag u of τ , we define X_u as the set of nodes of G contained in the bag and G_u as the subgraph of G induced by all nodes in all the bags descendant from u in τ (including u). We have $TW = \max_{u \in \tau} |X_u| - 1$. Without loss of generality, we consider that τ is a *nice tree decomposition*, meaning it can be rooted such that: if u is the root or a leaf of τ , then $|X_u| = 0$; if u has two children u_1 and u_2 , then $X_u = X_{u_1} = X_{u_2}$, we say u is a *join bag*; if u has one children u' then either there exists $v \in V$ such that $X_u = X_{u'} \cup \{v\}$, we say u is a *introduce bag* or there exists $v \in V$ such that $X_{u'} = X_u \cup \{v\}$, we say u is a *forget bag*; and finally no bag has three or more children.

It is possible to build, from an optimal decomposition, a nice decomposition that is also optimal, with $O(|V|)$ bags in linear time [5]. We use a classical dynamic programming algorithm to solve MTDW using the tree decomposition τ . Each bag u is associated with a set of states and each state is associated with a subproblem that can be solved recursively using the states of the children of u . In the following definitions, if $X \subset V$, $E(X)$ are the edges connecting X in E .

► **Definition 2 (States of a bag).** For each bag, we define a set $S(u)$ of states. A state of u contains m integers k_1, k_2, \dots, k_m , with $k_j \leq \max |C|$; an integer $c \leq n$; a subset $Y \subset X_u$; a subset $F \subset E(Y)$; a mapping d_1 associating $u \in Y$ to a non negative integer $d_1(v) \leq \min(d(v) - d_{G_u}(v), \Delta)$; a second mapping d_2 associating $v \in Y$ to a non negative integer $d_2(v)$ such that $\min(d_1(v) + d_F(v), \Delta) \leq d_2(v) \leq \min(d(v), \Delta)$ and a third mapping C associating $v \in Y$ to a positive integer $C(v) \in \llbracket 1; |Y| \rrbracket$ such that, if $(v, w) \in F$, then $C(v) = C(w)$ and such that the number of distinct values $C(v)$ for all the nodes $v \in Y$ is lower than c . We write $s = (u, k_1, k_2, \dots, k_m, c, Y, F, d_1, d_2, C)$.

► **Lemma 3.** $|S(u)| \leq (2 \max |C| + 1)^m \cdot n \cdot 2^{TW} \cdot 2^{TW^2} \cdot (\Delta + 1)^{2TW} \cdot TW^{TW}$

Proof. Let $s = \{u, k_1, k_2, \dots, k_m, c, Y, F, d_1, d_2, C\} \in S(u)$. Then $|k_j| \leq \max |C|$ and $c \leq n$, Y and F are subsets of X_u and $E(X_u)$, containing respectively at most TW and TW^2 elements, d_1 and d_2 associate a value between 0 and Δ to at most TW nodes and C associates a value lower than $|Y| \leq TW$ to at most TW nodes. ◀

► **Definition 4.** Let (FOR) be the following auxilliary problem: given a bag u and a state $s = (u, k_1, k_2, \dots, k_m, c, Y, F, d_1, d_2, C)$ of $S(u)$, we search for a forest f such that:

- (i) f is included in G_u ;
- (ii) for every $j \in \llbracket 1; m \rrbracket$, $\sum_{v \in G_u \setminus X_u} C_j(v, d_f(v)) \leq k_j$;
- (iii) f covers Y but not $X_u \setminus Y$;
- (iv) f contains every edge in F but no edge in $E(X_u) \setminus F$;
- (v) f contains c trees;
- (vi) for $v, w \in Y$, v and w are in the same tree of f if and only if $C(v) = C(w)$;
- (vii) for $v \in Y$, if $d_2(v) < \Delta$, $d_f(v) = d_2(v) - d_1(v)$ else $d_f(v) \geq d_2(v) - d_1(v)$.

If such a forest exists, we say f is a feasible solution of s and we set the cost of the forest as $\Omega(s, f) = \sum_{v \in Y} C_{m+1}(v, d_2(v)) + \sum_{v \in G_u \setminus Y} C_{m+1}(v, d_f(v))$. We search for the optimal forest $f^*(s)$ with minimum cost $\Omega^*(s) = \Omega(s, f^*(s))$. If no such forest exists, $\Omega^*(s) = +\infty$.

XX:4 An FPT Algorithm for subTree Problems Parameterized with the Treewidth.

136 We will then refer to Properties $s(\text{i}), s(\text{ii}), \dots, s(\text{vii})$ of a feasible solution of a state s , or
137 simply Properties (i), (ii), \dots (vii) if there is no ambiguity with the state.

138 2.1 Root

139 In order to solve MTDW, we have to search for the optimal forest of a state of the root. We
140 can easily check the following lemma:

141 ► **Lemma 5.** *Let r be the root of T . Then the optimal solution of the instance \mathcal{I} of MTDW
142 is $f^*(s)$ for $s = (r, K_1, K_2, \dots, K_m, 1, \emptyset, \emptyset, \{\}, \{\}, \{\})$.*

143 We now exhibit a recursive relation between each bag and its children to compute $f^*(s)$ and
144 $\Omega^*(s)$. This relation depends on the type of bag. In the next subsections, we start with the
145 termination point and then deal with the forget, introduce and join bags.

146 2.2 Leaves

147 ► **Lemma 6.** *Let u be a leaf bag of τ . Then if $s = (u, k_1, k_2, \dots, k_m, 0, \emptyset, \emptyset, \{\}, \{\}, \{\})$ then
148 the empty forest is feasible and optimal for s if $k_j \geq 0$ for every $j \in \llbracket 1; m \rrbracket$. In that case
149 $\Omega^*(s) = 0$. For any other state $s \in S(u)$, $\Omega^*(s) = +\infty$.*

150 **Proof.** If u is a leaf, then X_u and G_u are empty. Any feasible forest f of s is empty by
151 Property (i) and satisfies for every $j \in \llbracket 1; m \rrbracket$, $\sum_{v \in G_u \setminus X_u} C_j(v, d_f(v)) = 0$. Consequently,
152 any set with $k_j < 0$ for some j has no feasible solution by Property (ii) and has $\Omega^*(s) = +\infty$.
153 Any other state has only one feasible solution, the empty forest, of cost 0. ◀

154 2.3 Forget bags

155 Let u be a forget bag and u' be the child of u . Let x be the node forgotten by u : $X_{u'} = X_u \cup \{x\}$.
156 Let $s = (u, k_1, k_2, \dots, k_m, c, Y, F, d_1, d_2, C) \in S(u)$. We want to compute $\Omega^*(s)$.

157 ► **Definition 7.** \mathcal{P} is a set of parameters $(k'_1, k'_2, \dots, k'_m, F', d', C')$ such that $0 \leq d' \leq$
158 $\min(d(x), \Delta)$; for $j \in \llbracket 1; m \rrbracket$, $k_j - C_j(x, d') \geq -\max |C|$ and $k'_j = \min(k_j - C_j(x, d'), \max |C|)$;
159 for $v, w \in Y$, $C'(v) = C'(w) \Leftrightarrow C(v) = C(w)$; $F \subset F' \subset F \cup \gamma_Y(x)$; and for $v \in Y$, if there
160 exists a path connecting v to x with edges of F' , $C'(x) = C'(v)$.

161 Given a tuple $p = (k'_1, k'_2, \dots, k'_m, F', d', C') \in \mathcal{P}$, let $s'(p)$ be the following state:
162 $s'(p) = (u', k'_1, k'_2, \dots, k'_m, c, Y \cup \{v\}, F', d_1 \cup \{x \rightarrow 0\}, d_2 \cup \{x \rightarrow d'\}, C') \in S(u')$.
163 Finally, let $S' = \{(u', k_1, k_2, \dots, k_m, c, Y, F, d_1, d_2, C)\} \cup \{s'(p), p \in \mathcal{P}\}$.

164 ► **Lemma 8.** $\Omega^*(s) = \min_{s' \in S'} \Omega^*(s')$

165 **Proof.** Let f be any subforest of G_u . We first prove that f is a feasible solution of s if and
166 only if there exists $s' \in S'$ such that f is a feasible solution of s' , and that, in that case,
167 $\Omega(f', s) = \Omega(f', s')$. Either $x \in f$ or not. We consider the two cases.

168 The feasible solutions of the state $s' = (u', k_1, k_2, \dots, k_m, c, Y, F, d_1, d_2, C)$ are exactly the
169 feasible solutions of s not containing x . Thus, if x is not in f , then f is a feasible solution of
170 s if and only if f is a feasible solution of s' . Similarly, the formula of the cost of the solution
171 is identical in s and s' : $\Omega(s, f) = \Omega(s', f)$.

172 We now assume that $x \in f$. Let F' be F and the edges connecting x to the nodes of Y
173 in f . Let $d' = \min(d_f(x), \Delta)$. For every $j \in \llbracket 1; m \rrbracket$, we set $k'_j = \min(k_j - C_j(x, d'), \max |C|)$.
174 Finally, we define the mapping C' as $C'(v) = C(v)$ for every $v \in Y$ and $C'(x) = C(v)$ for

175 some arbitrary node $v \in Y$ that is in the same tree as x in f . If no such node exists, we set
 176 $C'(x) = |Y| + 1$. Clearly, $(k'_1, k'_2, \dots, k'_m, F', d', C')$ satisfies all the properties of a tuple of \mathcal{P}
 177 except possibly $k_j - C_j(x, d') \geq -\max|C|$.

178 We first show that if there exists $j \in \llbracket 1; m \rrbracket$ such that $k_j - C_j(x, d') < -\max|C|$
 179 then f is not feasible for s and f is not feasible for any state $s' \in S'$. Firstly, if f is
 180 feasible for s , then by Property $s(\text{ii})$, $\sum_{v \in G_u \setminus X_u} C_j(v, d_f(v)) \leq k_j$. As $C_j(x, d_f(x)) =$
 181 $C_j(x, d')$, $k_j - C_j(x, d') \geq \sum_{v \in G_u \setminus X_u \cup \{x\}} C_j(v, d_f(v)) \geq -\max|C|$. Secondly, we assume
 182 there exists a state $s'' = s'(k''_1, k''_2, \dots, k''_m, F'', d'', C'') \in S'$ such that f is feasible for
 183 s'' . As $(k''_1, k''_2, \dots, k''_m, F'', d'', C'') \in \mathcal{P}$, then $-\max|C| \leq k_j - C_j(x, d'')$. In addition,
 184 by Property $s''(\text{vii})$, $d_f(x) = d''_2(x) - d''_1(x) = d''$ if $d'' < \Delta$ or $d_f(x) \geq d''$ otherwise.
 185 However, we defined d' as $\min(d_f(x), \Delta)$. Consequently $d' = d''$. At last, by Property $s''(\text{ii})$,
 186 $-\max|C| \leq k_j - C_j(x, d'') = k_j - C_j(x, d')$.

187 We now assume that $k_j - C_j(x, d') \geq -\max|C|$ for every $j \in \llbracket 1; m \rrbracket$. We can then safely
 188 set $s' = s'(k'_1, k'_2, \dots, k'_m, F', d', C') \in S'$ and show that f is a feasible solution of s' if and
 189 only if f is feasible for s .

190 Properties $s(\text{i})$ and $s'(\text{i})$ are satisfied as f is, by hypothesis, a subforest of $G_u = G_{u'}$. As
 191 c is unchanged, $s(v)$ and $s'(v)$ are identical. We have Property $s(\text{iii})$ if and only if f covers Y
 192 but not $X_u \subset Y = X_{u'} \setminus Y \cup \{x\}$ if and only if we have Property $s'(\text{iii})$. Similarly $s(\text{iv})$ and
 193 $s'(\text{iv})$ are equivalent. Properties $s(\text{vi})$ and $s'(\text{vi})$ are equivalent by construction of C' .

194 We now consider Properties $s(\text{vii})$ and $s'(\text{vii})$. Let $d'_1 = d_1 \cup \{x \rightarrow 0\}$ and $d'_2 = d_2 \cup \{x \rightarrow$
 195 $d'\}$. As $d_1(v) = d'_1(v)$ and $d_2(v) = d'_2(v)$ for every node $v \in Y$, Property $s(\text{vii})$ is equivalent
 196 to Property $s'(\text{vii})$ restricted to Y . We finally show that Property $s'(\text{vii})$ is always true for
 197 the node x . Indeed, as $d'_2(x) - d'_1(x) = d'$, as $d'_2(x) = d'$ and as $d' = d_f(x)$ if $d_f(x) < \Delta$ and
 198 Δ otherwise, then $d_f(x) = d'_2(x) - d'_1(x)$ if $d'_2(x) < \Delta$ and $d_f(x) \geq d'_2(x) - d'_1(x)$ otherwise.

199 At last, we consider Properties $s(\text{ii})$ and $s'(\text{ii})$. For every $j \in \llbracket 1; m \rrbracket$, $C_j(x, d_f(x)) =$
 200 $C_j(x, d')$ whatever the value of d' is. Consequently $\sum_{w \in G_u \setminus X_u} C_j(v, d_f(v)) \leq k_j \Leftrightarrow$
 201 $\sum_{v \in G_{u'} \setminus X_{u'}} C_j(v, d_f(v)) \leq k_j - C_j(x, d_f(x)) \leq k_j - C_j(x, d')$.

202 In addition $\sum_{v \in G_u \setminus X_u \cup \{x\}} C_j(v, d_f(v)) \leq \max|C|$. Thus Properties $s(\text{ii})$ is equivalent to:
 203 for every $j \in \llbracket 1; m \rrbracket$, $\sum_{v \in G_{u'} \setminus X_{u'}} C_j(v, d_f(v)) \leq k'_j$.

204 As a conclusion, f is feasible for s if and only if f is feasible for s' . Moreover, an
 205 argument similar to the one of the previous paragraph can be used to prove that $\Omega(s, f) =$
 206 $\sum_{v \in Y} C_{m+1}(v, d_2(v)) + \sum_{v \in G_u \setminus Y} C_{m+1}(v, d_f(v)) = \sum_{v \in Y} C_{m+1}(v, d'_2(v)) + C_{m+1}(x, d') +$
 207 $\sum_{v \in G_{u'} \setminus Y \cup \{x\}} C_{m+1}(v, d_f(v)) = \Omega(s', f)$. Consequently, $\Omega^*(s) = \min_{s' \in S'} \Omega^*(s')$. ◀

208 2.4 Introduce bags

209 Let u be an introduce bag, u' be the child of u and x be the node introduced by u with
 210 $X_u = X_{u'} \cup \{x\}$. Let $s = (u, k_1, k_2, \dots, k_m, c, Y, F, d_1, d_2, C) \in S(u)$.

211 ► **Lemma 9.** *Let f be a feasible solution of s , then $\gamma_f(x) = \gamma_F(x)$.*

212 **Proof.** Recall that, in a tree decomposition, if a node belongs to two bags u_1 and u_2 ,
 213 it belongs to all the bags on the path connecting u_1 and u_2 . As $x \notin X_{u'}$, then x is
 214 not in any descendant bag of u' . Consequently, the only edges incident to x in G_u are
 215 $\gamma_{X_u}(v) = \{(x, v) | v \in X_u\}$. From the edges of $\gamma_{X_u}(v)$, we are only allowed to put $\gamma_F(x)$ in
 216 the forest f by Property (iv). ◀

217 Let $H = (Y, F)$ be the graph induced by the edges in F .

218 ► **Lemma 10.** *If $x \in Y$ and $d_1(x) + d_F(x) \neq d_2(x)$ then $\Omega^*(s) = +\infty$. If $x \in Y$ and x has
 219 no neighbor in H and there exists v such that $C(x) = C(v)$ then $\Omega^*(s) = +\infty$.*

XX:6 An FPT Algorithm for subTree Problems Parameterized with the Treewidth.

220 **Proof.** Let f be a feasible solution of s . By Lemma 9, if $d_F(x) < d_2(x) - d_1(x)$, then
 221 $d_f(x) < d_2(x) - d_1(x)$ and there is a contradiction with Property (vii). If $C(x) = C(v)$, in
 222 any feasible solution f of s , x and v are in the same tree by Property (vi). However, by
 223 Lemma 9, $\gamma_f(x) = \gamma_F(x)$. Thus, if $\gamma_F(x) = \emptyset$, there is a contradiction. ◀

224 We now assume the hypothesis of the previous lemma are false. We build a state s' of
 225 $S(u')$. We set $c' = c + d_F(x) - 1$ if $x \in Y$ and c otherwise; $Y' = Y \setminus \{x\}$; $F' = F \setminus \gamma(x)$; for
 226 every $v \in Y'$, $d'_1(v) = \min(\Delta, d_1(v) + 1)$ if $(x, v) \in F$ and $d_1(v)$ otherwise; and $d'_2(v)$ is $d_2(v)$.

227 We also build a mapping C' with the following procedure. If $v \notin Y$ or v has no neighbor
 228 in Y then C' is C restricted to Y' . Otherwise, we build a sorted list $L = [a_1, a_2, \dots, a_{|L|}]$
 229 containing the elements of $\{C(v) | v \in Y'\}$. For every $v \in Y'$ such that $C(v) = a_i$, we
 230 set $C'(v) = i$. We then arbitrarily order the new connected components of H obtained
 231 by removing x as $\mathcal{C} = [\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_{|\mathcal{C}|}]$. For every node $v \in \mathcal{C}_j$ for $j \in \llbracket 2; |\mathcal{C}| \rrbracket$, we reset
 232 $C'(v) = |\mathcal{C}| + j - 1$. We can easily check the following lemma:

233 ▶ **Lemma 11.** C' maps every node of Y' to an integer between 1 and $|Y'|$ such that
 234 ■ if $C(v_1) \neq C(v_2)$, then $C'(v_1) \neq C'(v_2)$;
 235 ■ if $C(v_1) = C(v_2)$ and v_1 and v_2 are connected by a path containing x in H if and only if
 236 $C'(v_1) \neq C'(v_2)$.

237 We finally define $s' = (u', k_1, k_2, \dots, k_m, c', Y', F', d'_1, d'_2, C')$.

238 ▶ **Lemma 12.** If $x \notin Y$, then $\Omega^*(s) = \Omega^*(s') + C_{m+1}(x, 0)$.

239 **Proof.** Let f be any subforest of G_u . We first show that f is a feasible solution of s if and
 240 only if f is a feasible solution of s' .

241 If $x \notin Y$ then $s' = (u', k_1, k_2, \dots, k_m, c, Y, F, d_1, d_2, C)$. Thus, Properties $s(\text{ii})$, $s(\text{v})$, $s(\text{vi})$
 242 and $s(\text{vii})$ are identical to Properties $s'(\text{ii})$, $s'(\text{v})$, $s'(\text{vi})$ and $s'(\text{vii})$.

243 If f contains x , then f satisfies neither Properties $s(\text{iii})$ nor Property $s'(\text{i})$ thus is not
 244 feasible for s and s' . If, on the contrary, f does not contain x , then, Properties $s(\text{i})$ and $s'(\text{i})$
 245 are satisfied, and Properties $s(\text{iii})$ and $s(\text{iv})$ are equivalent to $s'(\text{iii})$ and $s'(\text{iv})$.

246 Consequently, the feasible forests of s are feasible for s' and conversely. In addition,
 247 if f is feasible (and thus does not contain x), we have $\Omega(s, f) = \sum_{v \in Y} C_{m+1}(v, d_2(v)) +$
 248 $\sum_{G_u \setminus Y} C_{m+1}(v, d_f(v)) + C_{m+1}(x, d_f(x)) = \Omega(s', f) + C_{m+1}(x, 0)$ and the lemma follows. ◀

249 ▶ **Lemma 13.** If $x \in Y$, then $\Omega^*(s) = \Omega^*(s') + C_{m+1}(x, d_2(x))$.

250 **Proof.** We consider two sets: \mathcal{F} are the subforests of G_u satisfying Properties $s(\text{iii})$ and $s(\text{iv})$
 251 ; and \mathcal{F}' are the subforests of $G_{u'}$ satisfying Properties $s'(\text{iii})$ and $s'(\text{iv})$. Note that, firstly,
 252 any other forest is respectively not a feasible solution of s or s' ; secondly, that from any
 253 subforest $f \in \mathcal{F}$ we can obtain a subforest of \mathcal{F}' by removing x and every incident edge to x
 254 ; thirdly, that from any subforest $f' \in \mathcal{F}'$ we can obtain a subforest of \mathcal{F} by adding x and
 255 $\gamma_F(x)$; and lastly, by Lemma 9, that the two previous transformations are opposite and
 256 describe a bijection between \mathcal{F} and \mathcal{F}' .

257 Let then $f \in \mathcal{F}$ and $f' \in \mathcal{F}'$ be two associated forests. We now show that f is feasible for
 258 s if and only if f' is feasible for s' . Firstly, by definition of \mathcal{F} and \mathcal{F}' , f satisfy Properties $s(\text{i})$,
 259 $s(\text{iii})$ and $s(\text{iv})$ and f' satisfies Property $s'(\text{i})$, $s'(\text{iii})$ and $s'(\text{iv})$.

260 We have Property $s(\text{v})$ for f if and only if f has c trees. During the transformation
 261 process from f to f' , the tree containing x is replaced by $d_F(x)$ new trees, one for each
 262 incident edge of x in f . Consequently f has c trees if and only if f' has $c + d_F(x) - 1 = c'$
 263 trees if and only if Property $s'(\text{v})$ is satisfied by f' .

264 If f satisfies Property $s(\text{vi})$, then two nodes v_1 and v_2 of Y' are in the same tree in f' if
 265 and only if they were in the same tree in f and were not connected through x if and only if
 266 $C''(v_1) = C''(v_2)$ by Lemma 11. We now assume f' satisfies Property $s'(\text{vi})$. Two nodes v_1
 267 and v_2 of Y' are in the same tree in f if and only if the trees containing v_1 and v_2 in f' are
 268 the same or are connected by x in f' if and only if $C(v_1) = C(v_2)$ by Lemma 11. We finally
 269 consider x . By Lemma 10, either x has no neighbor in H in which case $C(x) \neq C(v)$ for
 270 every node $v \in Y$ or x has a neighbor w in H in which case, by Definition 2, $C(x) = C(w)$.
 271 In the first case, by Lemma 9, the tree of x in f contains only that node, and Property $s(\text{vi})$
 272 is satisfied. In the second case, as Property $s(\text{vi})$ is true for every nodes $v_1, v_2 \in Y'$, then a
 273 node v is in the tree containing w (and x) if and only if $C(v) = C(w) = C(x)$.

274 We now deal with Properties $s(\text{vii})$ and $s'(\text{vii})$. Recall first that we considered a case
 275 where Lemma 10 cannot be applied, meaning that $d_1(x) + d_F(x) = d_2(x)$. By Lemma 9,
 276 $d_f(x) = d_F(x) = d_2(x) - d_1(x)$. Thus, Properties $s(\text{vii})$ is true for x . We then just have
 277 to check that the two properties are equivalent for every node in Y' . We separate two
 278 cases depending if the node is a neighbor of x or not in F . Let $v \in Y'$ be a neighbor
 279 of x in F . In that case $d_f(v) = d_{f'}(v) + 1$. If $d_1(v) < \Delta$, then $d'_1(v) = d_1(v) + 1$.
 280 Consequently, $d_f(v) = d_2(v) - d_1(v) \Leftrightarrow d_{f'}(v) = d'_2(v) - d'_1(v)$ and $d_f(v) \geq d_2(v) - d_1(v) \Leftrightarrow$
 281 $d_{f'}(v) \geq d'_2(v) - d'_1(v)$. As $d_2(v) = d'_2(v)$, Properties $s(\text{vii})$ and $s'(\text{vii})$ are equivalent for
 282 the node v in that case. If now $d_1(v) = \Delta$, then $d'_1(v) = d'_2(v) = d_2(v) = \Delta$. Thus
 283 $d_2(v) - d_1(v) = d'_2(v) - d'_1(v) = 0 \leq d_{f'}(v) \leq d_f(v)$. So the Properties are true for v . Let
 284 finally $v \in Y'$ which is not a neighbor of x . In that case $d_f(v) = d_{f'}(v)$, $d_2(v) = d'_2(v)$ and
 285 $d_1(v) = d'_1(v)$, thus the equivalence is true for v .

286 We end with Properties $s(\text{ii})$ and $s'(\text{ii})$. For every node $v \in G_u \setminus X_u$, by Lemma 9, v is
 287 not a neighbor of x in f . Thus $d_f(v) = d_{f'}(v)$. In addition, $G_{u'} \setminus X_{u'} = G_u \setminus X_u$, thus the two
 288 properties are identical.

289 As a conclusion, f is feasible for s if and only if f' is feasible for s' . In addition,
 290 if f is feasible for s (and thus contains x), we have $\Omega(s, f) = \sum_{v \in Y'} C_{m+1}(v, d_2(v)) +$
 291 $C_{m+1}(x, d_2(x)) + \sum_{G_{u'} \setminus Y'} C_{m+1}(v, d_f(v)) = \Omega(s', f) + C_{m+1}(x, d_2(x))$. ◀

292 2.5 Join bags

293 Let u be a join bag and u' and u'' be the two children of u . We recall that $X_u = X_{u'} = X_{u''}$.

294 ▶ **Lemma 14.** $G_{u'} \cap G_{u''} = X_u$

295 **Proof.** Let $v \in G_{u'} \cap G_{u''}$. Then v is contained in a descendant bag of u' in the tree
 296 decomposition τ and in a descendant bag of u'' . Consequently, it belongs to every bag on
 297 the path linking those two descendants, including u . Thus $v \in X_u$. ◀

298 Let $s = (u, k_1, k_2, \dots, k_m, c, Y, F, d_1, d_2, C) \in S(u)$. We want to compute $\Omega^*(s)$. Given a
 299 mapping C , we write $\#C$ as the number of distinct values in the image of C .

300 ▶ **Definition 15.** \mathcal{Q} is a set of parameters $(k'_1, k'_2, \dots, k'_m, k''_1, k''_2, \dots, k''_m, c', c'', d'_1, d''_1, C', C'')$
 301 such that $|k'_j|, |k''_j| \leq \max |C|$; $k''_j = \min(\max |C|, k_j - k'_j)$; $d_1(v) \leq d'_1(v) \leq d_2(v)$, $d''_1(v) =$
 302 $\max(d'_1(v) - d_F(v), 0)$; $C(v_1) = C(v_2)$ if and only if there exists a list $(x_1 = v_1, x_2, \dots, x_p =$
 303 $v_2) \in Y$ such that for all $i \in \llbracket 1; p-1 \rrbracket$, $C'(x_i) = C'(x_{i+1})$ or $C''(x_i) = C''(x_{i+1})$; and
 304 $c' + c'' - \#C' - \#C'' = c - \#C$.

305 Given a tuple $q = (k'_1, k'_2, \dots, k'_m, k''_1, k''_2, \dots, k''_m, c', c'', d'_1, d''_1, C', C'') \in \mathcal{Q}$,
 306 let $s'(q) = (u', k'_1, k'_2, \dots, k'_m, c, Y, F, d_1, d'_1, C') \in S(u')$
 307 and $s''(q) = (u'', k''_1, k''_2, \dots, k''_m, c'', Y, F, d'_1, d_2, C'') \in S(u'')$.

308 ► **Lemma 16.** *Let f be a feasible solution of s and let $f' \subset G_{u'}$ and $f'' \subset G_{u''}$ obtained by
 309 respectively removing $(G_{u''} \setminus X_u)$ and $(G_{u'} \setminus X_u)$ from f . There exists $q \in \mathcal{Q}$ such that f' is
 310 feasible for $s'(q)$ and f'' is feasible for $s''(q)$.*

311 **Proof.** We first build the tuple q . For every $j \in \llbracket 1; m \rrbracket$, we set $k'_j = \sum_{v \in G_{u'} \setminus X_u} C_j(v, d_f(v))$
 312 and $k''_j = \min(\max |C|, k_j - k'_j)$. For every node $v \in Y$, we set $d'_1(v) = \min(d_{f'}(v) + d_1(v), \Delta)$
 313 and $d''_1(v) = \max(d'_1(v) - d_F(v), 0)$. We set c' as the number of trees in f' and C' such that
 314 for any two nodes $v_1, v_2 \in Y$, $C'(v_1) = C'(v_2) \Leftrightarrow v_1$ and v_2 are in the same tree of f' . We
 315 similarly set c'' and C'' . Hereinafter, we demonstrate that $q \in \mathcal{Q}$.

316 Indeed, $|k_j| \leq \max |C|$ and $k''_j = \min(\max |C|, k_j - k'_j)$ by definition. By Property $s(\text{ii})$,
 317 $\sum_{v \in G_{u'} \setminus X_u} C_j(v, d_f(v)) \leq k_j$. As $\sum_{v \in G_{u''} \setminus X_u} C_j(v, d_f(v)) = \sum_{v \in G_{u'} \setminus X_u} C_j(v, d_f(v)) -$
 318 $\sum_{v \in G_{u'} \setminus X_u} C_j(v, d_f(v)) \leq k_j - k'_j$, $-\max |C| \leq k_j - k'_j$, then $|k''_j| \leq \max |C|$.

319 Let $v \in Y$. As $d_{f'}(v) \geq 0$ and $d_1(v) \leq \Delta$, $d'_1(v) \geq d_1(v)$. By definition, $d''_1(v) =$
 320 $\max(d'_1(v) - d_F(v), 0)$. By Property $s(\text{vii})$, if $d_2(v) < \Delta$, then $d_1(v) + d_f(v) = d_2(v) < \Delta$.
 321 As $d_f(v) \geq d_{f'}(v)$, $d_1(v) + d_{f'}(v) < \Delta$ and then $d'_1(v) = d_1(v) + d_{f'}(v)$. Consequently,
 322 $d''_1(v) + d_{f'}(v) \leq d_2(v)$. If $d_2(v) = \Delta$, then either $d_1(v) + d_{f'}(v) < \Delta$ and then $d'_1(v) \leq d_2(v)$
 323 or $d_1(v) + d_{f'}(v) \geq \Delta$ and then $d'_1(v) = \Delta = d_2(v)$.

324 We finally have to prove the two last properties of \mathcal{Q} . Let $G' = (G_{u'} \setminus X_u)$ and $G'' =$
 325 $(G_{u''} \setminus X_u)$. The difference $c - \#C$ (resp. $c' - \#C'$ and $c'' - \#C''$) is the number of trees
 326 in f (resp. f' and f'') not containing any node in Y . By Lemma 14, $G' \cap G'' = \emptyset$. Thus
 327 $c - \#C = c' - \#C' + c'' - \#C''$.

328 Let now v_1 and v_2 be two nodes of X_u . Then $C(v_1) = C(v_2)$ if and only if v_1 and
 329 v_2 are in the same tree. There exists a path $P = (p_1 = v_1, p_2, \dots, p_{|P|} = v_2)$ connecting
 330 v_1 and v_2 in that tree. Let x_1, x_2, \dots, x_p be the $p \geq 2$ nodes of $P \cap Y$. For each couple
 331 (x_i, x_{i+1}) , either x_i and x_{i+1} are connected by an edge in F , then $C'(x_i) = C'(x_{i+1})$ and
 332 $C''(x_i) = C''(x_{i+1})$; or x_i and x_{i+1} are connected by a subpath of P consisting of nodes of
 333 G' or G'' . Thus x_i and x_{i+1} are either in the same tree in f' or in f'' , which is equivalent to
 334 $C'(x_i) = C'(x_{i+1}) \vee C''(x_i) = C''(x_{i+1})$ by definition of C' and C'' .

335 Consequently, $q \in \mathcal{Q}$ and we can safely define $s' = s'(q)$ and $s'' = s''(q)$. Firstly by
 336 definition, f, f' and f'' respectively satisfy $s(\text{i})$, $s'(\text{i})$ and $s''(\text{i})$. In addition, by Lemma 14,
 337 and because $X_u = X_{u'} = X_{u''}$, the properties (iii) and (iv) of s, s' and s'' are equivalent.
 338 Properties $s'(\text{v})$, $s'(\text{vi})$, $s''(\text{v})$ and $s''(\text{vi})$ are satisfied by definition of c', C', c'' and C'' .

339 We now focus on Properties $s'(\text{vii})$ and $s''(\text{vii})$. Let $v \in Y$. If $d'_1(v) = d_1(v) + d_{f'}(v)$,
 340 then Property $s'(\text{vii})$ is satisfied. If $d'_1(v) = \Delta$ then $d'_1(v) = \Delta \leq d_1(v) + d_{f'}(v)$ and the
 341 property is also proven. We now have to prove that $d_{f''}(v) = d_2(v) - d''_1(v)$. Note firstly that
 342 $d_f(v) = d_{f'}(v) + d_{f''}(v) - d_F(v)$ because $d_{f'}$ and $d_{f''}$ count the edges in F twice.

343 ■ If $d_2(v) < \Delta$, then, by Property $s(\text{vii})$, $d_f(v) = d_2(v) - d_1(v)$. In addition, $d'_1(v) \leq d_2(v) <$
 344 Δ by definition of \mathcal{Q} , then $d_1(v) = d_1(v) + d_{f'}(v)$. As $d_{f'}(v) \geq d_F(v)$ by Property $s(\text{iii})$,
 345 then $d'_1(v) - d_F(v) \geq 0$ and $d''_1(v) = \max(d'_1(v) - d_F(v), 0) = d'_1(v) - d_F(v)$. Finally,
 346 $d_{f''}(v) = d_f(v) - d_{f'}(v) + d_F(v) = d_2(v) - d''_1(v)$.

347 ■ If $d_2(v) = \Delta$, then $d_f(v) \geq d_2(v) - d_1(v)$.
 348 ■ If $d'_1(v) = d_1(v) + d_{f'}(v)$ then $d_{f''}(v) = d_f(v) - d_{f'}(v) + d_F(v) \geq d_2(v) - d_1(v) -$
 349 $d'_1(v) + d_1(v) + d_F(v)$. As $d''_1(v) \leq d'_1(v) - d_F(v)$, $d_{f''}(v) \geq d_2(v) - d''_1(v)$.
 350 ■ If $d'_1(v) = \Delta$ then $d''_1(v) = \max(\Delta - d_F(v), 0)$. If $d''_1(v) = \Delta - d_F(v)$ then $d_2(v) - d''_1(v) =$
 351 $d_F(v) \leq d_{f''}(v)$ by Property $s(\text{iii})$. If $d''_1(v) = 0$ then $\Delta - d_F(v) \leq 0$. In addition,
 352 $d_2(v) - d''_1(v) = \Delta \leq d_F(v) \leq d_{f''}(v)$.

353 Consequently Property $s''(\text{vii})$ is satisfied.

354 We end with Properties $s'(\text{ii})$ and $s''(\text{ii})$. The former is true by definition of k'_j . By
 355 Property $s(\text{ii})$, $\sum_{v \in G_{u'} \setminus X_u} C_j(v, d_f(v)) \leq k_j$. Consequently, $\sum_{v \in G_{u''} \setminus X_u} C_j(v, d_f(v)) =$

356 $\sum_{v \in G_u \setminus X_u} C_j(v, d_f(v)) - \sum_{v \in G_{u'} \setminus X_u} C_j(v, d_f(v)) \leq k_j - k'_j$. In addition, $\sum_{v \in G_{u''} \setminus X_u} C_j(v, d_f(v)) \leq$
 357 $\max |C|$, then $\sum_{v \in G_{u''} \setminus X_u} C_j(v, d_f(v)) \leq k'_j$.

358 As a consequence, f' and f'' are feasible solutions of the states s' and s'' . ◀

359 Due to lack of space, the proof of the converse property, given in the following lemma,
 360 can be found in Appendix B. The used arguments are similar to the ones in the proof of
 361 Lemma 16.

362 ▶ **Lemma 17.** *Let $q \in \mathcal{Q}$ and f' (respectively f'') be a feasible solution of $s'(q)$ (respectively*
 363 *$s''(q)$). Then $f = f' \cup f''$ is feasible for s .*

364 ▶ **Lemma 18.** $\Omega^*(s) = \min_{q \in \mathcal{Q}} \Omega^*(s'(q)) + \Omega^*(s''(q)) - \sum_{v \in Y} C_{m+1}(v, d'_1(v)) - \sum_{v \in X_u \setminus Y} C_{m+1}(v, 0)$.

365 **Proof.** In the two lemmas 16 and 17, we have $f = f' \cup f''$ and f' (respectively f'') can
 366 be obtained by removing $(G_{u''} \setminus X_u)$ (respectively $(G_{u'} \setminus X_u)$) from f . We have $\Omega(s', f') =$
 367 $\sum_{v \in Y} C_{m+1}(v, d'_1(v)) + \sum_{v \in X_u \setminus Y} C_{m+1}(v, 0) + \sum_{v \in G_{u'} \setminus X_u} C_{m+1}(v, d_f(v))$ and $\Omega(s'', f'') =$
 368 $\sum_{v \in Y} C_{m+1}(v, d_2(v)) + \sum_{v \in X_u \setminus Y} C_{m+1}(v, 0) + \sum_{v \in G_{u''} \setminus X_u} C_{m+1}(v, d_{f''}(v))$. By Lemma 14,
 369 $\Omega(s, f) = \Omega(s', f') + \Omega(s'', f'') - \sum_{v \in Y} C_{m+1}(v, d'_1(v)) - \sum_{v \in X_u \setminus Y} C_{m+1}(v, 0)$. ◀

370 2.6 Main theorem

371 ▶ **Lemma 19.** *There exists an algorithm solving MTDW with time complexity*
 372 *$O(n^4 \cdot (m + TW^3) \cdot (2 \max |C| + 1)^{3m} \cdot 2^{3TW+3TW^2} \cdot (\Delta + 1)^{6TW} \cdot TW^{3TW})$.*

373 **Proof.** If we compute $f^*(s)$ and $\Omega^*(s)$ for $s = (r, K_1, K_2, \dots, K_m, 1, \emptyset, \emptyset, \{\}, \{\}, \{\})$, by
 374 Lemma 5, we get the result. We can recursively compute those values using Lemmas 6, 8,
 375 12, 13 and 18. Consequently, we can use a dynamic programming algorithm to solve the
 376 problem in polynomial time, for instance an iterative algorithm that iterate through the bags
 377 of τ using a reversed breadth-first search algorithm and apply the lemmas for every state
 378 of every bag. We recall that by Lemma 3, for every bag u , the number of state in $S(u)$ is
 379 bounded by $B = (2 \max |C| + 1)^m \cdot n \cdot 2^{TW} \cdot 2^{TW^2} \cdot (\Delta + 1)^{2TW} \cdot TW^{TW}$.

380 The time complexity of the calculation of $\Omega^*(s)$, for some state $s \in S(u)$, depends on
 381 the type of the bag u . For a leaf, the computation is done in time $O(m)$. If u is not a
 382 leaf, we assume that $\Omega^*(s')$ was computed for every state $s' \in S(u')$, for every child u' of u
 383 and is accessible in constant time. For a forget bag, the computation consists in building
 384 S' and computing $\min_{s' \in S'} \Omega^*(s')$. The first step can be done by enumerating the at most
 385 B states of u' . For each such state, using Definition 7 to check if it belongs to S' is done
 386 in time $O(m + TW^2)$. The complexity is then $O((m + TW^2) \cdot B)$. For an introduce bag,
 387 the computation first consists in checking the two properties of Lemma 10 in time $O(TW)$.
 388 Then a state $s' \in S(u')$ is then computed for Lemmas 12 and 13 in $O(m)$. Computing the
 389 minimum value is done in constant time. The complexity is then $O(TW + m)$. For a join
 390 bag, we similarly enumerate every couple of states of u' and u'' and check if the related
 391 parameters belongs to \mathcal{Q} . This last part is done in time $O(m + TW^3)$. The TW^3 term comes
 392 from the penultimate property of \mathcal{Q} that can (naively) be done by running Y^2 depth first
 393 searches in the nodes of Y . Every other property is checked in constant time, in $O(m)$ or in
 394 $O(TW)$. Thus, the complexity for that bag is in $O((m + TW^3) \cdot B^2)$.

395 As the number of bags in the tree decomposition τ is $O(|V|) = O(n)$, the total number of
 396 states we have to consider is $O(n \cdot B)$. The overall complexity is then $O(n \cdot (m + TW^3) \cdot B^3)$. ◀

397 From the time complexity of Lemma 19, we can immediately deduce Theorem 1.

398 **3 Hardness Result**

399 This section provides four hardness results to prove that Theorem 1 cannot be adapted when
 400 m is not fixed, when TW or Δ are neither fixed nor a parameter or when $\max |C|$ is binary.

401 **► Theorem 20.** *($m = 1, \Delta = 2$)-MTDW is NP-Hard, even if $\max |C|$ is unary.*

402 **Proof.** The Minimum Leaf Spanning Tree problem is NP-Hard and, as stated in Appendix A,
 403 can be expressed as a subproblem of MTDW where $m = 1, \Delta = 2$ and $\max |C| = n$. ◀

404 **► Theorem 21.** *($m = 0$)-MTDW is W[1]-Hard with respect to TW , even if $\max |C|$ is unary.*

405 **Proof.** We give an FPT-reduction from the General Factors problem in which, given an
 406 undirected graph $H = (V_H, E_H)$ and, for each node $v \in V_H$, a subset $\beta(v) \subset \llbracket 1; d(v) \rrbracket$, we
 407 search for a subset $F \subset E_H$ such that, for each node $v \in V_H$, the number of edges of F
 408 incident to v is in $\beta(v)$. Such a subset is called a β -factor of H . GF is W[1]-hard with respect
 409 to the treewidth of H [9].

410 Given an instance $\mathcal{I} = (H = (V_H, E_H), \beta)$ of General Factors with treewidth TW , we
 411 build an instance $\mathcal{J} = (G, C_1)$ of MTDW as follows. From the graph H , we build the graph
 412 G by adding one node s to G and by replacing each edge $e = (u, v) \in E_H$ by a path of 5
 413 nodes u, e_u, e_s, e_v, v . We then link s to every node of V_H and to every node e_s for $e \in E_H$.

414 C_1 is the following function: for each node $v \in V_H$, then $C_1(v, d) = 0$ if $d - 1 \in \beta(v)$ and
 415 1 otherwise ; for each edge $e \in E_H$, $C_1(e_s, d) = 0$ if $d = 1$ or $d = 3$ and 1 otherwise; for
 416 each edge $e = (u, v) \in E_H$, $C_1(e_u, 1) = C_1(e_v, 1) = 0$ and $C_1(e_u, d) = C_1(e_v, d) = 1$ for every
 417 $d \neq 1$; and $C_1(s, d) = 0$ if $d = |V_H| + |E_H|$ and 1 otherwise.

418 This reduction is done in polynomial time with respect to $|V_H| + |E_H|$. We now prove
 419 there exists an optimal solution for \mathcal{J} with cost at most 0 if and only if H has a β factor.

420 Let T be a tree where $C_1(v, d_T(v)) = 0$ for every node in T . Then $(u, e_u) \in T \Leftrightarrow (v, e_v) \in T$
 421 for all $e = (u, v) \in E_H$. Indeed, if we assume for instance that $(u, e_u) \in T$ and $(v, e_v) \notin T$,
 422 then $(e_v, e_s) \in T$ otherwise e_v would have degree 0 in T and the cost of T would not be 0.
 423 Similarly, $(e_u, e_s) \notin T$, thus (e_s, s) cannot be in T as the degree of e_s should be either
 424 1 or 3. Finally (e_s, s) is necessarily in T as all the incident edges of s must be in T to get
 425 a tree with cost 0. Let then F be the edges $e \in E_H$ for which (u, e_u) and (v, e_v) are in T .
 426 The degree in T of a node u is the degree of u in F plus 1, as u is connected to s in T ; and
 427 as the cost of the tree is 0, then $d_T(u) - 1 = d_F(u) \in \beta(v)$. Thus there exists an optimal
 428 solution for \mathcal{J} with cost 0 if and only if H has a β factor.

429 On the other hand, given a β -factor F of H , by selecting all edges incident to s , (u, e_u)
 430 and (v, e_v) for $(u, v) \in F$ and (e_u, e_s) and (e_v, e_s) for $(u, v) \notin F$, we get a tree of cost 0.

431 Finally, the treewidth of G can be expressed as a fonction of the treewidth of H as it is
 432 at most $TW + 3 \cdot TW \cdot (TW - 1)/2 + 1$. Indeed, from a tree decomposition τ of H , we can
 433 build a tree decomposition of G by adding s to every node of τ and by adding e_u, e_s and e_v
 434 to every node of T containing u and v . Consequently there exists an FPT reduction from
 435 General Factors to MTDW. ◀

436 **► Theorem 22.** *($\Delta = 2, TW = 2$)-MTDW is NP-Hard and W[1]-Hard with respect to m ,
 437 even if $\max |C|$ is unary.*

438 **Proof.** We prove this result with an FPT reduction from the Partitioned Clique problem,
 439 parameterized with the size of the searched clique. Let $H = (V, E)$ be an undirected graph
 440 where V is partitioned into k independent sets $V = V_1 \cup V_2 \cup \dots \cup V_k$, the partitioned Clique

441 problem consists in the search for a clique of size k in H , containing one node in each set V_i .
 442 This problem is NP-Hard and W[1]-Complete with respect to k [6].

443 Given an instance (H, k) of the Partitioned Clique problem, we assume without loss
 444 of generality that every set V_i is of size s , and E_{ij} , the edges linking V_i and V_j , is of size
 445 $\sigma(ij)$. We set $V_i = (v_{i1}, v_{i2}, \dots, v_{is})$ and $E_{ij} = (e_{ij1}, e_{ij2}, \dots, e_{ij\sigma(ij)})$. We build an instance
 446 $\mathcal{I} = (G, C_1, C_2, \dots, C_{m+1}, K_1, K_2, \dots, K_m)$ of MTDW parameterized with m with $\Delta = 2$
 447 and $TW = 2$ as follows. We first add a star to G with a center x and $2k + k \cdot (k - 1)$
 448 leaves $\{w_i, w'_i, i \in \llbracket 1; k \rrbracket\} \cup \{f_{ij}, f'_{ij}, i < j \in \llbracket 1; k \rrbracket\}$. For each $i \in \llbracket 1; k \rrbracket$, we connect w_i and
 449 w'_i with a path P_i containing $2|V_i| + 2$ nodes $P_i = (w_i, v_{i1}, v'_{i1}, v_{i2}, v'_{i2}, \dots, v'_{is}, v_{is}, w'_i)$. For
 450 each $i < j \in \llbracket 1; k \rrbracket$, we connect f_{ij} and f'_{ij} with a path Q_{ij} containing $2|E_{ij}| + 2$ nodes
 451 $Q_{ij} = (f_{ij}, e_{ij1}, e'_{ij1}, e_{ij2}, e'_{ij2}, \dots, e_{ij\sigma(ij)}, e'_{ij\sigma(ij)}, f'_{ij})$. Note that G is a set of cycles with a
 452 common node x , and is thus outerplanar. Consequently, the treewidth of G equals 2.

453 We set $m = k \cdot (k - 1)$. In order to simplify the description, we first set $C_j(v, d) = 0$
 454 for every node v , degree d and constraint C_j . We then reset some of the values. For each
 455 $i < j \in \llbracket 1; k \rrbracket$, we build four constraints. For readability, we denote them by C_{ij}, C'_{ij}, C_{ji}
 456 and C'_{ji} . For every node $v_{ip} \in V_{H_i}$, we set $C_{ij}(v_{ip}, 1) = -C'_{ij}(v_{ip}, 1) = p$. For every node
 457 $v_r \in V_{H_j}$, we set $C_{ji}(v_r, 1) = -C'_{ji}(v_r, 1) = r$. For every edge $e_{ijq} = (v_p, v_r) \in E_{ij}$, we
 458 set $C_{ij}(e_{ijq}, 1) = -C'_{ij}(e_{ijq}, 1) = -p$ and $C_{ji}(e_{ijq}, 1) = -C'_{ji}(e_{ijq}, 1) = -r$. Finally, we
 459 set $K_{ij} = K'_{ij} = K_{ji} = K'_{ji} = 0$. The cost function C_{m+1} will imply a spanning tree
 460 constraint with some edge covering constraint: for $v \in V$, $C_m(v, 0) = 1$; for each node
 461 $v \in \{w_i, w'_i, f_{i,j}, f'_{i,j}\}$ for some i or (i, j) , $C_m(v, d < 2) = 1$. Note that $\Delta = 2$. We search for
 462 the existence of a feasible solution of cost at most 0.

463 We first characterize the properties of a feasible solution T of \mathcal{I} with cost 0. Due to
 464 the cost constraint C_{m+1} , every node must be spanned by T . In addition, for every node
 465 $v \in \{w_i, w'_i, f_{i,j}, f'_{i,j}\}$, v is of degree two in T . As a consequence, every edge incident to x is in
 466 T . Let now $i \in \llbracket 1; k \rrbracket$, as T is a spanning tree, exactly one edge of P_i must not be in T : exactly
 467 one node v_{ip} of P_i has degree 1 in T . We can similarly state that for every $j \in \llbracket i+1; k \rrbracket$, there
 468 exists $q \leq \sigma(ij)$ and $r \leq s$ such that $d_T(e_{ijq}) = d_T(v_{jr}) = 1$. Assuming $e_{ijq} = (v_{ia}, v_{jb})$ for
 469 some $a, b \leq s$, $\sum_{v \in V} C_{ij}(v, d_T(v)) = p - a$ and $\sum_{v \in V} C_{ji}(v, d_T(v)) = r - b$. As C_{ij}, C'_{ij}, K_{ij}
 470 and K'_{ij} are opposite numbers, we have $\sum_{v \in V} C_{ij}(v, d_T(v)) = 0$, thus $p = a$. Similarly, we
 471 have $r = b$. Consequently, there exists in H an edge linking v_{ip} and v_{jr} . Consequently, the
 472 set $\{v_{ip}, i \in \llbracket 1; k \rrbracket, p \in \llbracket 1; s \rrbracket | d_T(v_{ip}) = 1\}$ is a clique of size k in H .

473 Conversely, if C is a clique with $|C| = k$, we order the nodes of C . Without loss
 474 of generality, let $C = (v_{11}, v_{21}, \dots, v_{k1})$. Then, the subgraph $G \setminus (\{(v_{i1}, v'_{i1}), i \in \llbracket 1; k \rrbracket\} \cup$
 475 $\{(e_{ij1}, e'_{ij1}), i \in \llbracket 1; k \rrbracket, j \in \llbracket i+1; k \rrbracket\})$, where $e_{ij1} = (v_{i1}, v_{j1})$ is a feasible solution of cost 0.

476 This transformation is then an FPT reduction with respect to k and a polynomial
 477 reduction. Consequently, the theorem follows. ◀

478 ▶ **Theorem 23.** ($\Delta = 2, TW = 2, m = 2$)-MTDW is (weakly) NP-Hard.

479 **Proof.** We prove this result with a reduction, indirectly from the Partitioned Clique problem,
 480 by starting with the instance \mathcal{I} build in the proof of Theorem 22. From \mathcal{I} we build a new
 481 instance \mathcal{I}' with $m = 2$ but where $\max |C|$ is exponential.

482 We do not change the graph G . We have the same cost function C_{m+1} . However, the C_{ij}
 483 functions are merged into a single function C_1 and the functions C'_{ij} are merged into C_2 .

484 Let n be the number of nodes in the graph from the Partitioned Clique instance then
 485 $|C_{ij}(v, d)| \leq n$ and $|C_{ji}(v, d)| \leq n$. Let $\theta = 2n|G| + 1$. For every node $v \in G$ and integer $d \leq$
 486 $d(v)$, we set $C_1(v, d) = -C_2(v, d) = \sum_{i=1}^k \left(\sum_{j=i+1}^k (n + C_{ij}(v, d)) \cdot \theta^{ik+j} + (n + C_{ji}(v, d)) \cdot \theta^{k^2+ik+j} \right)$
 487 and $K_1 = -K_2 = \sum_{i=1}^k \left(\sum_{j=i+1}^k n|G| \cdot \theta^{i \cdot k+j} + n|G| \cdot \theta^{k^2+i \cdot k+j} \right)$.

XX:12 An FPT Algorithm for subTree Problems Parameterized with the Treewidth.

488 T be a feasible solution of \mathcal{I}' if and only if
489
$$\sum_{i=1}^k \left(\sum_{j=i+1}^k \sum_{v \in G} (n + C_{ij}(v, d_T(v))) \cdot \theta^{ik+j} + \sum_{v \in G} (n + C_{ji}(v, d_T(v))) \cdot \theta^{k^2+ik+j} \right) = K_1.$$

490 However, for all i and j , $0 \leq \sum_{v \in G} (n + C_{ij}(v, d)) \leq 2n|G| < \theta$ and $0 \leq \sum_{v \in G} (n +$
491 $C_{ji}(v, d)) \leq 2n|G| < \theta$. Thus, the above equality is satisfied if and only if, for every i, j , we
492 have $\sum_{v \in G} (n + C_{ij}(v, d_T(v))) = \sum_{v \in G} (n + C_{ji}(v, d_T(v))) = n|G|$, if and only if, for every
493 i, j , $\sum_{v \in G} C_{ij}(v, d_T(v)) = \sum_{v \in G} C_{ji}(v, d_T(v)) = 0$ if and only if T is feasible for \mathcal{I} . ◀

494 **4 Conclusion and future works**

495 This work gives an FPT algorithm for many covering tree problems with respect to the
496 treewidth. The algorithm interest is mainly theoretical as its complexity makes it unpractical.
497 This is not really a surprise considering the high level of generalization of MTDW. It gives
498 a basis that can be used to build faster FPT algorithm for every subproblem by taking
499 into account the particularities of that problem. In the same way, the hardness results may
500 also be used as a working base to build NP-Hardness or W[1]-hardness with respect to the
501 treewidth for subproblems which do not satisfy the requirements of Theorem 1.

502 Those results can be extended to capture other classes of optimization problems. Firstly
503 we could focus on the cyclomatic number, the size of a cycle basis, which is another distance
504 between a graph and a tree. It would secondly be interesting to extend the results to other
505 classical covering structures like forests, matchings, paths and cliques. A last possible future
506 work would be to generalize the constraints. For instance, we could allow C_j to take as input
507 a node v and a subset of $\gamma_G(v)$ instead of a degree. Or instead of having $\sum_{v,d} C_j(v, d) \leq K_j$
508 for every j , we could have constraint such as $\min_{v,d} C_j(v, d) \leq K_j$.

509 **References**

- 510 1 S Arnborg, J Lagergren, and D Seese. Easy problems for tree-decomposable graphs. *Journal*
511 *of Algorithms*, 12(2):308–340, 1991. doi:10.1016/0196-6774(91)90006-K.
- 512 2 Markus Chimani, Petra Mutzel, and Bernd Zey. Improved Steiner tree algorithms for bounded
513 treewidth. In *Journal of Discrete Algorithms*, volume 16, pages 67–78, 2012. doi:10.1016/j.
514 jda.2012.04.016.
- 515 3 M Cygan, FV Fomin, L Kowalik, D Lokshtanov, D Marx, Ma Pilipczuk, Mi Pilipczuk,
516 and S Saurabh. *Parameterized Algorithms*. Springer, Cham, 2015. doi:10.1007/
517 978-3-319-21275-3.
- 518 4 RG Downey and MR Fellows. *Parameterized complexity*. Springer-Verlag New York, 1999.
519 doi:10.1007/978-1-4612-0515-9.
- 520 5 Ton Kloks. *Treewidth: computations and approximations*. Springer-Verlag Berlin Heidelberg,
521 1994. doi:10.1007/BFb0045375.
- 522 6 Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet shortest common
523 supersequence and longest common subsequence problems. *Journal of Computer and System*
524 *Sciences*, 67(4):757–771, 2003. doi:10.1016/S0022-0000(03)00078-3.
- 525 7 R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning
526 trees - Short or small. *SIAM Journal on Discrete Mathematics*, 9(2):178–200, 1996. doi:
527 10.1137/S0895480194266331.
- 528 8 Neil Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width.
529 *Journal of Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.
- 530 9 Marko Samer and Stefan Szeider. Tractable cases of the extended global cardinality constraint.
531 *Constraints*, 16(1):1–24, 2011. doi:10.1007/s10601-009-9079-y.

A Subproblems of MTDW

532

533 MTDW can be seen as a generalization of many covering tree problems in undirected graph.
 534 This appendix gives a non exhaustive list of such subproblems; how to rewrite them as a set
 535 of MTDW instances and what are the consequences of Theorem 1 on that problem.

536 ■ The Minimum Leaf Spanning Tree problem consists, given an undirected graph in the
 537 search for a spanning tree with a minimum number of leaves. We set $m = 1$. The
 538 constraint C_1 is a spanning tree constraint: $C_1(v, 0) = 1$, $C_1(v, d \geq 1) = 0$ and $K_1 = 0$,
 539 every node must be spanned. Note that the connectivity constraint is given by the fact
 540 that any feasible solution is a tree. The cost function C_2 counts the number of leaves:
 541 $C_2(v, 1) = 1$ and $C_2(v, d \neq 1) = 0$.

542 The treewidth of the graph is unchanged. We have $\max |C| = n$ and $\Delta = 2$. Consequently,
 543 due to Theorem 1, this problem is FPT with respect to the treewidth. Similarly, the
 544 Maximum Leaf Spanning Tree problem (in which the number of leaves is maximized) is
 545 FPT with respect to the treewidth. The sole difference is that $C_2(v, 1) = -1$ instead of 1.

546 ■ Another similar subproblem is the Minimum Branch Vertices problem, in which we search
 547 for a spanning tree with a minimum number of nodes with degree 3 or more. In that case,
 548 we set $C_2(v, d \leq 2) = 0$ and $C_2(v, d \geq 3) = 1$. It is then also FPT with respect to the
 549 treewidth as the treewidth is unchanged and as $\max |C| \leq n^2$ and $\Delta = 3$. If we consider
 550 the generalized version, in which we minimize the number of nodes of degree k or more,
 551 then this problem is FPT with respect to the treewidth and k .

552 ■ The Steiner Tree problem may be rewritten as a subproblem of MTDW. In that problem,
 553 a subset X of nodes, called *terminals*, must be spanned. Each edge e is weighted with
 554 $\omega(e)$ and we search for a minimum-cost tree. We set $m = 1$. We first set $C_1(v, 0) = 1$,
 555 $C_1(v, d > 0) = 0$ for every node $v \in X$ and $K_1 = 0$. A second step consists in modifying
 556 the graph in order to consider the weight of the edges. We split every edge $e = (u, v)$
 557 in two edges (u, v_e) and (v_e, v) and, we set $C_1(v_e, 0) = C_1(v_e, 2) = 0$, $C_1(v_e, 1) = 1$ to
 558 ensure that the edge cannot be partially used. We finally set $C_2(v_e, 2) = \omega(e)$.

559 The treewidth becomes the maximum of the treewidth and 3. Indeed, given a decompo-
 560 sition of the original graph, for each bag containing the two extremities u and v of an
 561 edge e , we attach to that bag another bag containing u , v and v_e . This new tree is a
 562 decomposition of the new graph. We have $\max |C| = |S| + n^2 \leq n + n^2$ (we recall that
 563 $\max |C|$ only takes into account the constraints and not the cost function) and $\Delta = 2$.
 564 Thus, Theorem 1 is a way to prove the following existing result [2]: the Steiner Tree
 565 problem is FPT with respect to the treewidth.

566 Similarly, it is also possible to prove that the Prize Collecting Steiner Tree problem is
 567 FPT with respect to the treewidth. Note that this is also an existing result [2]. Each
 568 edge e is weighted with $\omega(e)$ that must be paid if e belongs to the solution and each node
 569 v is weighted with a penalty $\pi(v)$ that must be paid if v does not belong to the solution.
 570 We handle the edges weight as in the Steiner Tree problem. We set $C_2(v, 0) = \pi(v)$ and
 571 $C_2(v, d > 0) = 0$ for every node v .

572 ■ The k -Minimum Spanning Tree problem, in which we search for a minimum-cost spanning
 573 tree containing at least k nodes, can similarly be proven FPT with respect to the treewidth.
 574 Note that this result is already given in [7]. We set $m = 2$. The edges are split as in the
 575 Steiner Tree problem and handled with a constraint C_1 and the cost function C_3 . We
 576 add a second constraint C_2 : $C_2(v, 0) = 0$, $C_2(v, d > 0) = -1$ and $K_2 = -k$.

577 ■ In the Budget Steiner Tree problem with Profits, the edges are weighted with a function ω
 578 and a budget B is given. Each node v is also weighted with a revenue $r(v)$. The objective

XX:14 An FPT Algorithm for subTree Problems Parameterized with the Treewidth.

579 is to maximize the total revenue of the spanned nodes without exceeding the budget B
580 with weights of the edges in the solution. We set $m = 2$. As for the previous problems,
581 we handle the edges by splitting them. However, instead of using a constraint C_1 and
582 the cost function C_3 , we use the two constraints C_1 and C_2 . We set $C_2(v_e, 2) = \omega(e)$ and
583 $K_2 = B$ so that the budget is not exceeded. The cost function C_3 computes the revenue
584 of the solution with $C_3(v, 0) = 0$ and $C_3(v, d > 0) = -r(v)$.
585 Note that, in this problem, the encoding of $\max |C|$ depends on the encoding of ω : the
586 problem is FPT with respect to the treewidth if ω is unary. We conjecture that the proof
587 of Theorem 23 can be adapted to the case where ω is binary to prove that this problem
588 is NP-Hard even if the treewidth is 2.

B Proof of Lemma 17

590 We detail in this appendix the proof of Lemma 17.

591 ► **Lemma 17.** *Let $q \in \mathcal{Q}$ and f' (respectively f'') be a feasible solution of $s'(q)$ (respectively*
592 *$s''(q)$). Then $f = f' \cup f''$ is feasible for s .*

593 **Proof.** Properties (i), (iii) and (iv) are obviously satisfied.

594 We now show that f satisfies Property $s(\text{ii})$. Let $j \in \llbracket 1; m \rrbracket$. By Properties $s'(\text{ii})$ and
595 $s''(\text{ii})$, $\sum_{v \in G_{u'} \setminus X_u} C_j(v, d_{f'}(v)) \leq k'_j$ and $\sum_{v \in G_{u''} \setminus X_u} C_j(v, d_{f''}(v)) \leq k''_j \leq k_j - k'_j$. By
596 Lemma 14, $\sum_{v \in G_u \setminus X_u} C_j(v, d_f(v)) \leq k_{j'} + k_j - k_{j'} = k_j$.

597 By definition of \mathcal{Q} , for every two nodes v_1 and v_2 , $C(v_1) = C(v_2)$ if and only if there
598 exists a path $(x_1 = v_1, x_2, \dots, x_p = v_2)$ such that, for each $i < p$, $C'(x_i) = C'(x_{i+1})$ or
599 $C''(x_i) = C''(x_{i+1})$. By Properties $s'(\text{vi})$ and $s''(\text{vi})$, this is equivalent to claim that x_i and
600 x_{i+1} belongs to the same tree in f' or in f'' which means that x_i and x_{i+1} belongs to the
601 same tree in f . Thus Property $s(\text{vi})$ is satisfied.

602 The number of trees in f' and f'' are respectively c' and c'' . The number of trees not
603 containing nodes in X_u are respectively $c' - \#C'$ and $c'' - \#C''$. Consequently, by Lemma 14,
604 the number of trees in f not containing a node of X_u is $c' - \#C' + c'' - \#C''$ and by definition
605 of \mathcal{Q} , this equals $c - \#C$: the number of trees in f is c .

606 We end with Property $s(\text{vii})$. Let $v \in Y$. Note firstly that $d_f(v) = d_{f'}(v) + d_{f''}(v) - d_F(v)$
607 because $d_{f'}$ and $d_{f''}$ count the edges in F twice. In the following, we consider multiple nested
608 subcases: either $d_2(v) < \Delta$ or $d_2(v) = \Delta$; either $d''_1(v) = d'_1(v) - d_F(v)$ or $d''_1(v) = 0$; and
609 either $d'_1(v) < \Delta$ or $d'_1(v) = \Delta$.

610 ■ If $d_2(v) < \Delta$, then $d'_1(v) \leq d_2(v) < \Delta$ by definition of \mathcal{Q} . By Property $s'(\text{vii})$, $d_{f'}(v) =$
611 $d'_1(v) - d_1(v)$. Thus, as f' covers F by Property $s'(\text{iv})$, $d'_1(v) \geq d_F(v) + d_1(v) \geq d_F(v)$.
612 Then, by definition of \mathcal{Q} , $d''_1(v) = d'_1(v) - d_F(v)$. Finally, by Property $s''(\text{vii})$ $d_{f''}(v) =$
613 $d_2(v) - d''_1(v)$. Then $d_f(v) = d_2(v) - d''_1(v) + d'_1(v) - d_1(v) - d_F(v) = d_2(v) - d_1(v)$.

614 ■ We now assume that $d_2(v) = \Delta$. By Properties $s'(\text{vii})$ and $s''(\text{vii})$, $d_{f'}(v) \geq d'_1(v) - d_1(v)$
615 and $d_{f''}(v) \geq d_2(v) - d''_1(v)$. If $d''_1(v) = d'_1(v) - d_F(v)$, then $d_f(v) \geq d_2(v) - d''_1(v) +$
616 $d'_1(v) - d_1(v) - d_F(v) \geq d_2(v) - d_1(v)$. If $d''_1(v) = 0$, then $d'_1(v) - d_F(v) \leq 0$.

617 ■ If $d'_1(v) = \Delta$, then, $\Delta \leq d_F(v)$. As f covers F by Property $s(\text{iv})$, $d_f(v) \geq d_F(v) \geq$
618 $\Delta = d_2(v) \geq d_2(v) - d_1(v)$.

619 ■ If $d'_1(v) < \Delta$, then, by Definition 2, $\min(d_1(v) + d_F(v), \Delta) \leq d'_1(v) < \Delta$. Thus
620 $d'_1(v) \geq d_1(v) + d_F(v) \geq d_F(v)$. As $d'_1(v) - d_F(v) \leq 0$, the two values are equal.
621 Consequently, $d_f(v) \geq d_2(v) - d''_1(v) + d'_1(v) - d_1(v) - d_F(v) = d_2(v) - d_1(v)$.

622 As a consequence, Property $s(\text{vii})$ is satisfied by f and thus f is feasible for s . ◀