



HAL
open science

Algorithmic Height Compression of Unordered Trees

Farah Ben-Naoum, Christophe Godin

► **To cite this version:**

Farah Ben-Naoum, Christophe Godin. Algorithmic Height Compression of Unordered Trees. *Journal of Theoretical Biology*, 2016, 389, pp.237 - 252. 10.1016/j.jtbi.2015.10.030 . hal-01412215

HAL Id: hal-01412215

<https://hal.science/hal-01412215>

Submitted on 8 Dec 2016

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.

Algorithmic Height Compression of Unordered Trees

Farah Ben-Naoum^a, Christophe Godin^b

^a*Evolutionary Engineering and Distributed Information Systems Laboratory, Department of Computer Science, Djillali Liabes University of Sidi Bel Abbes, Algeria, farahbennaoum@yahoo.fr*

^b*INRIA Sophia-Antipolis Mediterranee, Virtual Plants project-team, UMR AGAP, Montpellier, France, christophe.godin@inria.fr.*

Abstract

By nature, tree structures frequently present similarities between their subparts. Making use of this redundancy, different types of tree compression techniques have been designed in the literature to reduce the complexity of tree structures. A popular and efficient way to compress a tree consists of merging its isomorphic subtrees, which produces a directed acyclic graph (*DAG*) equivalent to the original tree. An important property of this method is that the compressed structure (i.e. the *DAG*), has the same height as the original tree, thus limiting partially the possibility of compression. In this paper we address the problem of further compressing this *DAG* in height. The difficulty is that compression must be carried out on substructures that are not exactly isomorphic as they are strictly nested within each-other. We thus introduced a notion of quasi-isomorphism between subtrees, that makes it possible to define similar patterns along any given path in a tree. We then proposed an algorithm to detect these patterns and to merge them, thus leading to compressed structures corresponding to *DAGs* augmented with return edges. In this way, redundant information is removed from the original tree in both width and height, thus achieving minimal structural compression. The complete compression algorithm is then illustrated on the compression of various plant-like structures.

Keywords: Plants modeling, branching structures, self-nestedness, quasi-isomorphism, height redundancy.

1. Introduction

Many plants like trees exhibit complex branching structures. These structures contain large numbers of components (branches, leaves, flowers, etc.) whose organization in space may give a mixed feeling of both order and disorder. The impression of disorder often comes from the fact that components apparently do not follow any clear deterministic rule and that spatial distribution of organs do not show exact symmetries. However, most of the time, behind this semi-chaos, the same branching structures also show some order in their organization as many sub-structures look similar, hierarchies can be identified among branches and some symmetries in shapes, although none exact-ones, seem to exist. During decades, botanists have strived to identify rules characterizing this order despite the apparent disorder in plant architectures, introducing notions such as repetition, modularity, gradients, symmetry of branching structures, e.g. Troll (1937); Arber (1950); Halle et al. (1978); Harper et al. (1986); Barthelemy (1991). These rules, mainly of qualitative nature, gave key insights on how to address the notion of order in plant architectures.

In the early 70's, computational formalisms started to emerge to represent formally the architecture of branching structures, e.g. Lindenmayer (1971); Lindenmayer and Rozenberg (1972); Lindenmayer (1975); Honda (1971). These new conceptual tools made it possible to consider the analysis of branching systems organization from a new and quantitative perspective. Many approaches started from there to describe models as finite automata or grammars that could reproduce by simulation the branching organization of plants, e.g. Borchert and Honda (1984); Reffye et al. (1989); Prusinkiewicz and Lindenmayer (1990). Among them, L-systems emerged in the plant modeling community as the most commonly used paradigm -based on rewriting rules- to model branching system development (Prusinkiewicz (1998)). Most of these approaches started from the design of a formal model embedding biological assumptions, and were defined to produce algorithmically branching systems similar to those observed in real plants. By contrast, the reverse (inverse) problem, namely building algorithmically a computational finite representation (model) from given observed branching systems, was much less studied.

On fractal images this inverse problem was solved on two dimensional images by Barnsley (1988, 2006) using models of fractal geometry. However, these models rely on iterated function systems (IFS) which correspond

38 to purely geometrical transformations. In particular, they don't take into
39 account topology of the studied biological structures.

40 To account on inference, many approaches based on L-systems were also
41 developed and reviewed later on in (Ben-Naoum (2009)). However, due to the
42 complexity of the general problem, these different approaches make various
43 simplifying assumptions. These assumptions in general are not satisfactory
44 in the context of study of biological organisms such as plants structures.

45 One particularly important simplifying hypothesis consists of focusing on
46 the result of the developmental process without considering the intermediate
47 stages of development. This leads to a whole class of approaches in which
48 models are constructed to capture plant branching architecture in a minimal
49 way. A first approach was proposed by Viennot et al. (1989) based on a
50 statistical analysis of binary tree organization. Subtrees were characterized
51 by an integer representing their amount of asymmetry (the so called Holton-
52 Strahler -HS- number (Deussen and Lintermann (2002))). Then parameters
53 were inferred from data to describe statistically the probability of finding a
54 tree with HS number n knowing the HS number m of the parent tree. A
55 similar statistical but more general approach was later used by Durand et al.
56 (2005, 2007), based on Markovian processes to describe subtrees frequencies
57 at any given node. These inference approaches are aimed to capture the
58 statistical properties of tree organization. Viewed as compression techniques,
59 they provide approximated strategies to compress trees, relying on different
60 types of *a priori* assumptions (e.g. Markovian property). An exact and
61 deterministic approach was recently introduced in Godin and Ferraro (2010).

62 In this latter work, the authors addressed the problem of recognizing sim-
63 ilar, possibly nested, patterns in plant structures, and to compress them as
64 directed acyclic graphs (*DAGs*) in a reversible way. Trees were represented
65 as unordered labeled tree structures, where no order is assumed on the chil-
66 dren of any node of the trees. Trees whose compression are linear *DAGs*
67 define the class of self-nested trees, i.e. trees that have very high compress-
68 ibility. From this, a degree of self-nestedness could be defined for any tree
69 as the normalized edit-distance of this tree to the class of self-nested trees
70 embedding the original tree, and a polynomial-time algorithm was designed
71 to compute this distance.

72 The tree compressions techniques defined in Godin and Ferraro (2010)
73 are based on the merging of isomorphic subtrees. Any two subtrees that are
74 merged therefore result in a structure with the same height. As a result, the
75 compressed *DAG*, where all the isomorphic subtrees have been merged, has

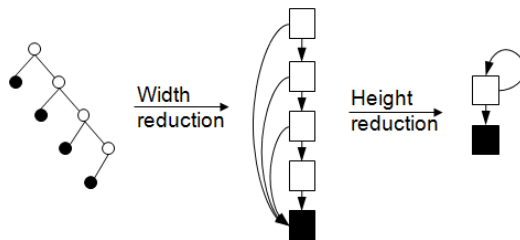


Figure 1: Basic example of width and height tree reduction

76 the same height as the original tree. If indeed compression occurs in width, no
 77 compression occurs in height. However, some trees such as high dichotomic
 78 trees or fish bone-like trees have a repetitive structure in height and are
 79 therefore poorly compressed by this technique. In this paper, we consider
 80 the problem of maximally compressing trees both in width and height with
 81 no loss of information. This idea is illustrated in Fig.1. In section 3, we recall
 82 the definition of tree reduction and self-nestedness introduced in Godin and
 83 Ferraro (2010). We then introduce and study in section 4 a weak extension
 84 of tree isomorphism, called quasi-isomorphism, that will make it possible
 85 to identify similar (but not identical) tree patterns at different heights of a
 86 tree. Based on these definitions and their properties, we present in section
 87 5 an algorithm able to compress in height the reduction of any tree. This
 88 algorithm is then applied to both theoretical tree-like structures and vegetal
 89 branching systems to characterize its compression ability.

90 For seek of clarity all property proofs described in the main text are given
 91 in the Online Supplementary Material.

92 2. Notational conventions

93 An *undirected graph*, is a pair $G = (V, E)$ where V denotes a finite set
 94 of vertices and E a finite set of unordered pairs of vertices called edges. A
 95 *complete graph* is a simple undirected graph in which every pair of distinct
 96 vertices is connected by a unique edge. A *clique* in an undirected graph
 97 $G = (V, E)$ is a subset of the vertex set $C \subseteq V$, such that the subgraph
 98 induced by C is complete (in some cases, the term clique may also refer to
 99 the subgraph). A *maximal clique* is a clique that cannot be extended by
 100 including one more adjacent vertex from the original set of vertices.

101 A *directed graph*, is a pair $G = (V, E)$ where all edges correspond to
 102 ordered pairs of vertices. Let (x, y) be an edge in E , x is called a *parent*

103 of y and y is a *child* of x . A vertex that has no child is called a *leaf*. We
 104 will denote, in the sequel, by $child(x)$ the set of all the children of x , and
 105 $parent(x)$ the set of all parents vertices of x . In a directed graph G , the
 106 *Indegree* of a vertex v , denoted $deg^-(v)$, is the number of its parents, and
 107 its *Outdegree*, denoted $deg^+(v)$, is the number of its children. The Indegree
 108 and the Outdegree of G are respectively the maximum Indegree and the
 109 maximum Outdegree of all G vertices.

110 A *path* from a vertex x to a vertex y is a (possibly empty) sequence of
 111 edges $\{(x_i, x_{i+1})\}_{i=1, M-1}$ such that $x_1 = x$, $x_M = y$, and M correspond to
 112 $|P|$ the *length* of P . For a path P we denote by V_P the set of all vertices
 113 that belong to P . In an other hand, $P = \lambda$ if P is an empty path. Where
 114 we denote by λ the empty path. Analogously, we say that there are no path
 115 between x and y , or there is an empty path.

116 A vertex x is called an *ancestor* of a vertex y (and y is called a *descendant*
 117 of x), noted $x \prec y$, if there exists a path from x to y . Analogically, for the
 118 edges $e = (x, y)$ and $e' = (x', y')$ we can say that $e \prec e'$ iff $y \prec x'$.

119 Given two paths $P_1 = \{(x_i, x_{i+1})\}_{i=1, N-1}$ and $P_2 = \{(y_j, y_{j+1})\}_{j=1, M-1}$:

- 120 • for $x_N = y_1$ we define the paths *union*
- 121 $P_1 \cup P_2 = \{(x_1, x_2), \dots, (x_{N-1}, x_N), (y_1, y_2), \dots, (y_{M-1}, y_M)\}$
- 122 • we denote $V_{P_1} \cap V_{P_2}$ the vertices paths *intersection* i.e. the subset of
- 123 vertices common to both P_1 and P_2
- 124 • we say that
 - 125 – $P_1 \subset P_2$ if the sequence made by the subset of edges common to
 - 126 both P_1 and P_2 correspond to P_1 (path *inclusion*)
 - 127 – $P_1 \prec P_2 \Leftrightarrow x_N \prec y_1$
 - 128 – $P_1 \circ P_2 \Leftrightarrow x_N = y_1$ (paths *succession*).

129 In a directed graph, we call *cycle* a non empty path whose extremities coin-
 130 cide to the same vertex. If the *path is elementary*, i.e. does not pass twice
 131 through the same vertex, it is called *elementary cycle*.

132 A *directed acyclic graph (DAG)* is a graph containing no cycle (but which
 133 may contain undirected cycles), (Preparata and Yeh (1973)). A *linear DAG*
 134 is a *DAG* containing at least one path that goes through all its vertices.

135 A *tree* T is a connected graph containing neither directed of undirected
 136 cycle. A *rooted tree* is a tree such that there exists a unique vertex, called

137 the root, which has no parent vertex. In a tree, each vertex, different from
 138 the root, has exactly one parent vertex. In the following, a rooted tree is
 139 simply called a tree.

140 In this paper, we consider *rooted unordered trees*, meaning that the order
 141 among the sibling vertices of any given vertex is not significant.

142 The degree *deg* of a tree is the maximum number of children of a vertex
 143 of T . The notation $|T|$ represents the number of T vertices.

144 The height $h(x)$ of a vertex x in a *DAG* is the length of the longest path
 145 from x to a leaf. The height $h(D)$ of a *DAG* D is the height of its root
 146 vertex, and its width, denoted $l(D)$, is the maximum number of vertices of
 147 the same height in all the *DAG*.

148 A *subtree* is a particular connected subgraph of a tree. Let x be a vertex
 149 of a tree $T = (V, E)$, $T[x]$ is a *complete subtree* if it is the maximal subtree
 150 rooted in x with:

151 $T[x] = (V[x], E[x])$, where $V[x] = \{y \in V/x \text{ is the ancestor of } y\}$ and
 152 $E[x] = \{(u, v) \in E/u \in V[x], v \in V[x]\}$. In the sequel, we will only consider
 153 complete subtrees and use the simpler term "subtree".

154 Let us consider two trees, $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$. A bijection ϕ
 155 from V_1 to V_2 is a *tree isomorphism* if for each $(x, y) \in E_1$, $(\phi(x), \phi(y)) \in E_2$.
 156 If there exists an isomorphism between T_1 and T_2 , the two structures are thus
 157 identical up to a relabeling of their components. In this case we say that T_1
 158 is *isomorphic to* T_2 .

159 A *multi-set* is a set of typed elements such that the number of elements
 160 of each type is known. It is defined as a set of pairs $M = \{(k, n_k)\}_k$ where
 161 k varies over the element types and n_k is the number of occurrences of type
 162 k in the set.

163 3. Tree reduction

164 3.1. Definition

165 We consider here the principle of *tree reduction* as was defined by Godin
 166 and Ferraro (2010). The aim of the *reduction* is to transform a rooted un-
 167 ordered tree T into a *directed acyclic graph* noted $D = (V, E)$. Each vertex of
 168 the D corresponds to an equivalence class of T subtrees. All subtrees belong-
 169 ing to the same class are *isomorphic*. All *DAG* vertices (classes) connected
 170 by edges are ordered by the *ancestor* partial order relation, noted \preceq , between
 171 the subtrees which they represent. In other words, let A and B two *DAG*
 172 vertices where $A \preceq B$, then the tree of class B is isomorphic to a subtree of

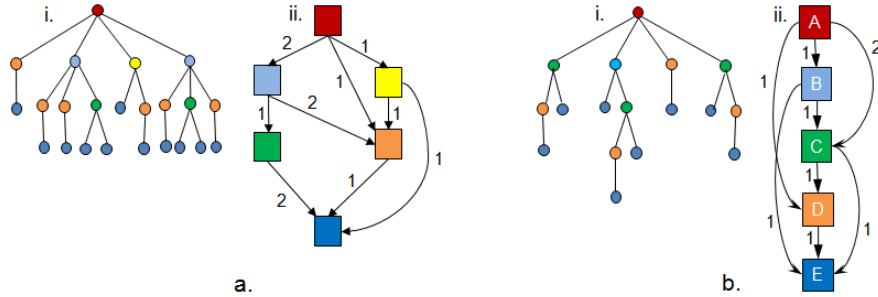


Figure 2: a.i. a tree T , and ii. the associated reduction graph $\mathcal{R}(T)$, b.i. the tree T_1 , and ii. its reduction graph $\mathcal{R}(T_1)$

173 the tree of class A . Each DAG contains a single root vertex corresponding to
 174 the class of the entire tree T and one terminal vertex representing the class
 175 of all T leaves.

176 To obtain a graph equivalent to the tree T , additional information is
 177 added in the DAG edges, as *edges weights*. For any edge (c_i, c_j) , its weight
 178 $n(c_i, c_j)$ is defined by the number of occurrences of the subtree of class c_j
 179 in this of class c_i . The obtained graph is called the reduction graph of T ,
 180 and noted by $\mathcal{R}(T) = (V, E)$, where E is a multi-set whose elements have the
 181 form $((c_i, c_j), n(c_i, c_j))$. The construction of the reduction graph of a tree can
 182 be carried out in time $O(|T|^2 deg(T) \log deg(T))$ (Godin and Ferraro (2010)).
 183 Fig.2.a shows an example of a tree and its reduction graph.

184 In the following sections we will consider $DAGs$ corresponding to reduc-
 185 tion graphs, i.e. $DAGs$ whose edges are augmented by weights. \mathcal{D} denotes
 186 the class of all $DAGs$.

187 3.2. Self-nested trees

188 Let us define the set of *self-nested* trees, as the trees in which all subtrees
 189 with identical height are isomorphic. Godin and Ferraro (2010) showed that
 190 self-nested-trees have the following characterizing properties:

- 191 • any two subtrees are either isomorphic or included one into another
 192 (one is a subtree of the other).
- 193 • their reduction $\mathcal{R}(T)$ is a linear DAG .

194 The reduction of the tree in Fig.2.b.i is the linear DAG depicted in Fig.2.b.ii.
 195 By contrast, the tree of Fig.2.a.i is not self-nested as its reduction is a non
 196 linear DAG (Fig.2.a.ii).

197 Based on these definitions, the degree of self-nestedness of any tree T can
 198 be quantified by computing the distance between T and the smallest self-
 199 nested tree (noted *NEST*) that embeds it. This distance is null if the tree is
 200 in the class of self-nested tree and augments as the tree contains increasingly
 201 self-nested structures. Godin and Ferraro (2010) showed that this distance
 202 and the corresponding *NEST* of a tree T can be computed in polynomial
 203 time by a *DAG* linearization algorithm.

204 An important property of both the reduction graph and the *NEST* of a
 205 tree is that they preserve the height of nodes in the original tree. This means
 206 that vertices in either the exact (the reduction graph) or the approximated
 207 ones (the *NEST*) corresponding to vertices in the original tree have exactly
 208 the same height. In Fig.2.b for example, the green node in the reduction
 209 graph has a height of 3 (Fig.2.b.ii), corresponding exactly to the height of
 210 the subtrees that it refers to in the original tree rooted also in green vertices
 211 (Fig.2.b.i). The same property is true for all the other colors. As a conse-
 212 quence, a tree of height N will necessarily have a compression with a number
 213 of vertices greater than N . In many cases, when N is of the same order of
 214 magnitude as the tree size $|T|$. This is clearly a limitation of the previous
 215 contraction procedures.

216 We are therefore lead to study how to compress trees in height as well as
 217 in width. For this, we need to relax the definition of isomorphism between
 218 trees to capture the notion of repetition of tree patterns independently of
 219 their height.

220 4. Quasi-periodic paths in a reduction graph

221 4.1. Intuitive idea of height reduction

222 Intuitively, a tree can be reduced in height if there are subtrees repeated
 223 in some way along some paths from the root to the leaves. But how to define
 224 exactly such repetitions?

225 Consider for instance tree of Fig.3.a. We see that the structure including
 226 the subtrees originating at vertex B are repeated at different heights of the
 227 original tree. The series of strictly nested subtrees $A(3) \subset A(2) \subset A(1) \subset$
 228 $A(0)$ are similar, but not exactly isomorphic. Actually, if we discard the right-
 229 hand side subtree of $A(i)$, $i = 0, \dots, 3$ (illustrated in Fig.3.b), on the picture,
 230 we remark that the remaining trees are indeed isomorphic. We shall say
 231 that these trees are quasi-isomorphic, i.e. isomorphic for almost all of their
 232 immediate subtrees except one, and shall show that nested quasi-isomorphic

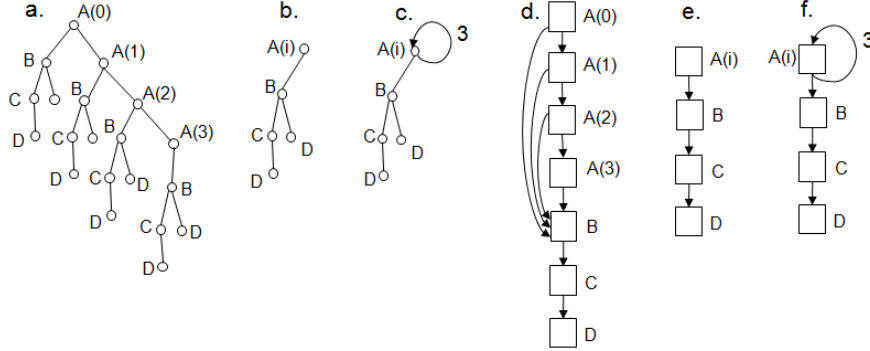


Figure 3: a. a height regular tree T , b. the repeated pattern of T , c. Intuitive elimination of T pattern repetitions, d. the DAG $\mathcal{R}(T)$, e. the sub DAG representing the regular repeated pattern, f. The intuitive height reduced DAG of $\mathcal{R}(T)$

233 sub-trees can be compressed by suppressing their repetition in the original
 234 tree and replacing them by loops over the repeated structure, as seen in the
 235 compressed tree of Fig.3.c. The objective is then to apply this principle of
 236 height compression directly to the associated DAG , as illustrated in graphs
 237 of Figs.3.d, 3.e and 3.f.

238 4.2. Vertex and edge signatures

239 **Definition 1 (Vertices signature)** Let $\mathcal{R}(T) = (V, E)$ a reduction graph.
 240 We associate with V the matrix $\sigma \in \mathbf{N}^{|V| \times |V|}$ of vertex signature, where each
 241 element σ_{xy} is defined as:

$$\sigma_{xy} = \begin{cases} n(x, y) & \forall ((x, y), n(x, y)) \in E \\ 0 & \text{else where} \end{cases}$$

242 In the matrix σ , the line associated with vertex x defines the signature σ_x of
 243 x . It represents the links of x with all its children vertices, and then describes
 244 the subtree $T[x]$.

245 **Example 1** Let $\mathcal{R}(T_1) = (V, E)$ the reduction graph of Fig.2.b.ii, where:
 246 $V = \{A, B, C, D, E\}$ and
 247 $E = \{((A, B), 1), ((A, C), 2), ((A, D), 1), ((B, C), 1), ((B, E), 1), ((C, D), 1),$
 248 $((C, E), 1), ((D, E), 1)\}$

249 The associated vertex signature matrix is given as follow:

$$\begin{array}{r}
\sigma_A = [\begin{array}{ccccc} A & B & C & D & E \\ 0 & 1 & 2 & 1 & 0 \end{array}] \\
\sigma_B = [\begin{array}{ccccc} 0 & 0 & 1 & 0 & 1 \end{array}] \\
\sigma_C = [\begin{array}{ccccc} 0 & 0 & 0 & 1 & 1 \end{array}] \\
\sigma_D = [\begin{array}{ccccc} 0 & 0 & 0 & 0 & 1 \end{array}] \\
\sigma_E = [\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \end{array}]
\end{array}$$

251 **Property 1** $\forall x, y \in V, \sigma_x \neq \sigma_y$.

252 **Definition 2 (Atomic path)** We define an atomic path in a reduction
253 graph as a path of length 1.

254 $\langle x, y \rangle$ denotes the atomic path from a vertex x to a vertex y .

255 **Definition 3** Given a vertices signature matrix σ , and given x, y two ver-
256 tices where $\sigma_{xy} \neq 0$. We define the vector $\sigma_{x,y}$ as the vector σ_x with σ_{xy} is
257 set to 0.

258 The vector $\sigma_{x,y}$ describes the constitution of the subtree associated with the
259 vertex x up to the subtree structure of its child y . This leads to the following
260 property.

261 **Property 2** $\sigma_{x,y} = \sigma_{x',y'} \Leftrightarrow x$ is isomorphic to x' up to a child node y in x
262 and a child node y' in x' .

263 Property 2 sets up the basis of a new relation which we call *quasi-isomorphism*
264 between $T[x]$ and $T[x']$.

265 **Example 2** Given two trees rooted in X and X' , illustrated respectively in
266 Fig.4.a and 4.c, and associated with the vertices x and x' in the corresponding
267 reduction graphs of Fig.4.b and 4.d. Note that the subtrees rooted in Y and
268 Y' are distinct. We obtain:

$$\left. \begin{array}{l} \sigma_{a_1} = \sigma_{a_2} \\ \sigma_{xa_1} = \sigma_{x'a_2} = 1 \\ \sigma_{b_1} = \sigma_{b_2} \\ \sigma_{xb_1} = \sigma_{x'b_2} = 1 \end{array} \right\} \xLeftrightarrow{Def.3} \sigma_{x,y} = \sigma_{x',y'}$$

269 Then X and X' are quasi-isomorphic sub-trees regarding their respective
270 descendants Y and Y' .

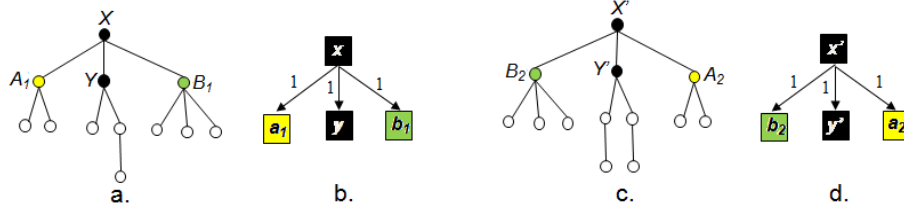


Figure 4: a. and c. quasi-isomorphic trees $T[X]$ and $T[X']$, b. and d. the corresponding reduction graphs $\mathcal{R}(T[X])$ and $\mathcal{R}(T[X'])$. Black vertices display their differences

271 **Definition 4 (Edge signature)** *The signature of an edge (x, y) , denoted*
 272 *by $\sigma_{x|y}$, is the line $\sigma_{x,y}$ augmented by the additional value of σ_{xy} on the right*
 273 *side.*

274 **Example 3** The edges signatures associated with the graph $\mathcal{R}(T_1)$ of Ex-
 275 ample 1 are given as follow:

$$\begin{aligned}
 276 \quad \sigma_A &= [0 \ 1 \ 2 \ 1 \ 0], \\
 277 \quad \sigma_{A,B} &= [0 \ 0 \ 2 \ 1 \ 0] \text{ and } \sigma_{AB} = 1 \text{ then:} \\
 278 \quad \sigma_{A|B} &= [0 \ 0 \ 2 \ 1 \ 0 \ 1] \text{ likewise} \\
 \sigma_{A|C} &= [0 \ 1 \ 0 \ 1 \ 0 \ 2] \\
 \sigma_{A|D} &= [0 \ 1 \ 2 \ 0 \ 0 \ 1] \\
 \sigma_{B|C} &= [0 \ 0 \ 0 \ 0 \ 1 \ 1] \\
 279 \quad \sigma_{B|E} &= [0 \ 0 \ 1 \ 0 \ 0 \ 1] \\
 \sigma_{C|D} &= [0 \ 0 \ 0 \ 0 \ 1 \ 1] \\
 \sigma_{C|E} &= [0 \ 0 \ 0 \ 1 \ 0 \ 1] \\
 \sigma_{D|E} &= [0 \ 0 \ 0 \ 0 \ 0 \ 1]
 \end{aligned}$$

280 The edges signatures allow us to study the quasi-isomorphism relation in
 281 a detailed and formal way on the basis of tree reduction graphs.

282 4.3. Quasi-isomorphism of paths

283 4.3.1. Quasi-isomorphism of atomic paths

284 **Definition 5 (Quasi-isomorphic atomic paths)** *In a reduction graph,*
 285 *given two distinct atomic paths $\langle x, y \rangle, \langle x', y' \rangle$. We say that $\langle x, y \rangle$ and $\langle x', y' \rangle$*
 286 *are quasi-isomorphic, noted $\langle x, y \rangle \cong \langle x', y' \rangle$, if and only if $\sigma_{x|y} = \sigma_{x'|y'}$.*

287 **Example 4 A.** From Example 2:

$$\left. \begin{aligned}
 288 \quad \sigma_{x,y} &= \sigma_{x',y'} \\
 \sigma_{xy} &= \sigma_{x'y'} = 1
 \end{aligned} \right\} \begin{aligned}
 &\xleftrightarrow{\text{Def.4}} \sigma_{x|y} = \sigma_{x'|y'} \\
 &\xleftrightarrow{\text{Def.5}} \langle x, y \rangle \cong \langle x', y' \rangle
 \end{aligned}$$

289 **B.** Given the set of edges signature computed in Example 3. There exist
 290 in this set, two equal edges signatures: $\sigma_{B|C} = \sigma_{C|D}$, which induce that
 291 $\langle B, C \rangle \cong \langle C, D \rangle$. This appears clearly in Fig.2.b.i in which the subtree
 292 associated with B except its child C , and the subtree associated with C
 293 except its child D are isomorphic.

294 **Property 3 (Fundamental property)** *Let e, f be two atomic paths start-*
 295 *ing from x and let e', f' be two atomic paths starting from x' , then*

$$((e \cong e') \wedge (f \cong f')) \Rightarrow ((f = e) \wedge (f' = e'))$$

296 In other terms, at most one pair of quasi-isomorphic atomic paths can start
 297 from a pair of vertices x and x' .

298 4.3.2. Quasi-isomorphism of paths

299 Let's consider two paths P and Q , where $P = \{e_1, \dots, e_n\}$, and $Q =$
 300 $\{f_1, \dots, f_m\}$.

301 **Definition 6 (quasi-isomorphic paths)** *We say that P is quasi-isomor-*
 302 *phic to Q (noted $P \cong Q$) if and only if $n = m$ and $e_i \cong f_i$ ($1 \leq i \leq n$).*

303 In other words, two paths are quasi-isomorphic if and only if their atomic
 304 paths are quasi-isomorphic when compared piecewise in the paths order.

305 **Example 5 A.** In the DAG $\mathcal{R}(T_1)$ (Fig.2.b) they are two quasi-isomorphic
 306 paths: $P = \langle B, C \rangle$ and $P' = \langle C, D \rangle$ from $\sigma_{B|C} = \sigma_{C|D}$.

307 **B.** In the DAG $\mathcal{R}(T_2)$ (Fig.5.a) they are two quasi-isomorphic paths: $P \cong P'$
 308 with $P = \{\langle B, E \rangle, \langle E, I \rangle\}$, $P' = \{\langle C, F \rangle, \langle F, H \rangle\}$ from $\sigma_{B|E} = \sigma_{C|F} =$
 309 $[000100000001]$, and $\sigma_{E|I} = \sigma_{F|H} = [000000100001]$.

310 **Notation:** A path $P = \{\langle x_1, x_2 \rangle, \dots, \langle x_{n-1}, x_n \rangle\}$ can be denoted as: $P =$
 311 $x_1 P x_n$ or $P = x_1 P_1 x_k P_2 x_n$ and so on. If $P = x_1 P_1 x_i P_i x_{i+1} P_n x_n$ and $|P_i| = 1$
 312 we can note $P = x_1 P_1 x_i x_{i+1} P_n x_n$.

313 4.3.3. Quasi-periodic path

314 A quasi-periodic path is a path decomposable in strictly smaller sub-paths
 315 which are all quasi-isomorphic with each other, and which are not themselves
 316 strictly decomposable.

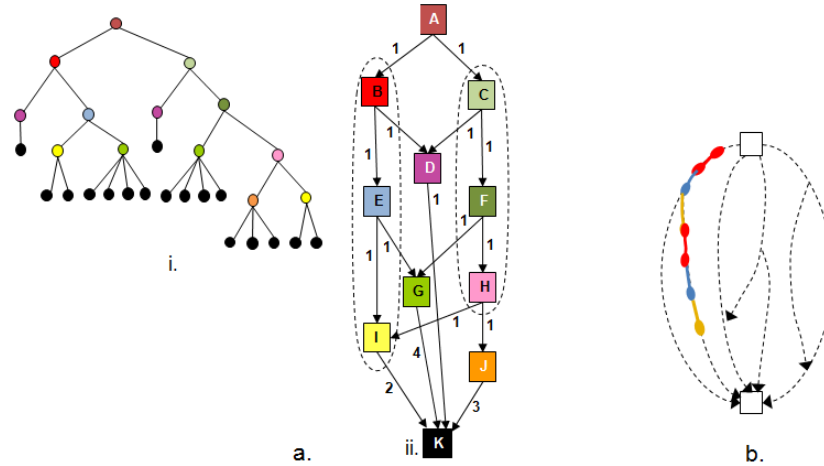


Figure 5: a.i. A tree T_2 , ii. its reduction $\mathcal{R}(T_2)$. We surround by dashed lines the two QIP s $P = \{\langle B, E \rangle, \langle E, I \rangle\}$ and $P' = \{\langle C, F \rangle, \langle F, H \rangle\}$, b. Illustration of QPP in a pattern of a DAG where the same color is used for all quasi-isomorphic atomic paths

317 **Definition 7 (quasi-periodic path QPP)** Given two vertices x, y with
 318 $x \prec y$. We say that xPy is a quasi-periodic path if there exist paths P_1, \dots, P_n
 319 ($P_i \neq \lambda$) and ($n \geq 2$) for which:

- 320 • $P = P_1 \cup \dots \cup P_n$
- 321 • $P_1 \cong \dots \cong P_n$
- 322 • $\forall i \leq n, P_i$ is not a quasi-periodic path.

323 P is called $QPP(x, y)$, and the sequence P_1, \dots, P_n the *normal decomposition*
 324 of P .

325 **Property 4** The normal decomposition of P is unique.

326 **Example 6 A.** From the graph $\mathcal{R}(T_1)$ of Fig.2.b, the path $P = P_1 \cup P_2$ with
 327 $P_1 = \langle B, C \rangle, P_2 = \langle C, D \rangle$ is a QPP .

328 **B.** Moreover, in $\mathcal{R}(T_2)$ of Fig.5.a.ii, although the paths P and P' are two
 329 quasi-isomorphic paths, they do not form a QPP since they are not consec-
 330 utive.

331 By definition, each QPP is a set of path chunks which are:

- 332 • embedded in the same global path,
- 333 • all quasi-isomorphic with each other,
- 334 • pairwise connected.

335 Therefore a *QPP* can be considered as a periodic path made by the repe-
 336 tition of quasi-isomorphic paths forming its normal decomposition. This is
 337 illustrated in Fig.5.b that shows a *QPP* whose normal decomposition is made
 338 up two subsequences. One sequence is indicated by red, blue and yellow.

339 Let us consider quasi-periodic paths P and Q with identical extremities.

340 **Property 5** *Let $xPy = QPP(x, y)$ and $xQy = QPP(x, y)$ then $P = Q$.*

341 The property shows that these two paths coincide.

342 **Property 6** *Every subsequence of a normal decomposition of a *QPP* is a*
 343 *normal decomposition of a smaller *QPP*.*

344 Let P_1, \dots, P_n be a normal decomposition of a *QPP* P , the path corre-
 345 sponding to a subsequence P_i, \dots, P_{i+k} ($i < n, 1 < k < n - i$) of the normal
 346 decomposition of P is itself a *QPP*.

347 4.3.4. Maximal Quasi-periodic path

348 **Definition 8 (Maximal quasi-periodic path *MQPP*)** *A maximal*
 349 *quasi-periodic path (noted *MQPP*) is a *QPP* maximum for path inclusion.*

350 Let P_1, \dots, P_n be the normal decomposition of a *MQPP* P , then there is no
 351 *QPP* Q ($P \subset Q$) such that P_1, \dots, P_n is a part of the normal decomposition
 352 of Q .

353 Let us denote by $MQPP(\mathcal{R}(T))$ the set of all the *MQPPs* of a reduction
 354 graph $\mathcal{R}(T) = (V, E)$.

355 **Property 7** *The set $MQPP(\mathcal{R}(T))$ can be computed in time $O(|E|^2 h(\mathcal{R}(T))$*
 356 *$l(\mathcal{R}(T)))$.*

357 4.3.5. Relative positions of maximal quasi-periodic paths in a reduction graph

358 Let us now study the relative positions of two *MQPPs* in a reduction
 359 graph. As for any two paths in a reduction graph, two *MQPPs* can either
 360 *intersect* or *not*, intersecting *MQPPs* can either be *nested* or *not*.

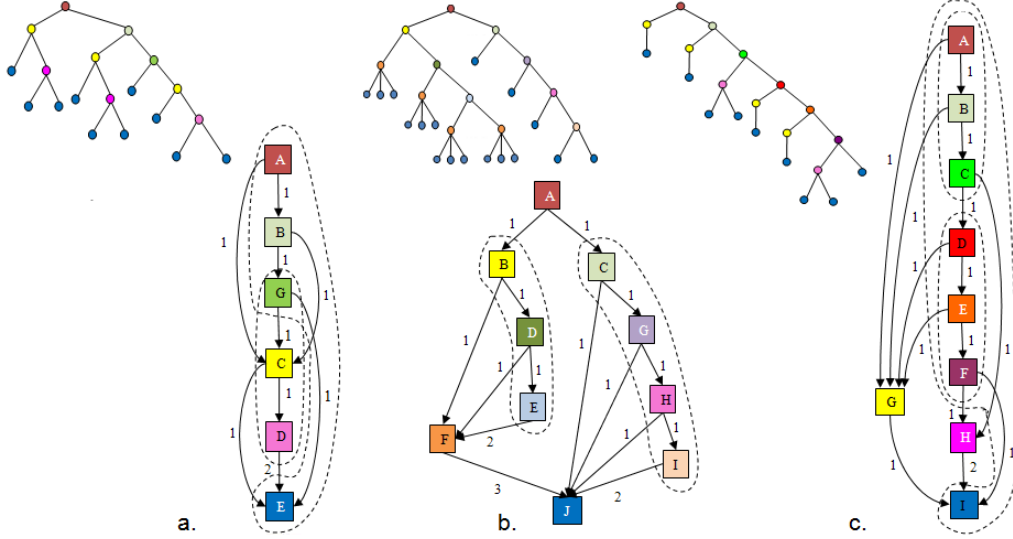


Figure 6: a. Tree T_3 and its reduction $\mathcal{R}(T_3)$ with two intersecting MQPPs. b. Tree T_4 and its reduction $\mathcal{R}(T_4)$ with two disjoint MQPPs. c. Tree T_5 and its reduction $\mathcal{R}(T_5)$ with a set of nested MQPPs

361 **Example 7 A.** In the reduction graph $\mathcal{R}(T_3)$ of Fig.6.a, the paths
 362 $P = \{\langle A, B \rangle, \langle B, G \rangle, \langle G, E \rangle\}$ and $P' = \{\langle G, C \rangle, \langle C, D \rangle\}$ are intersecting
 363 MQPPs. They have a common vertex G .

364 **B.** In the reduction graph $\mathcal{R}(T_4)$ of Fig.6.b, the paths $P = \{\langle B, D \rangle, \langle D, E \rangle\}$
 365 and $P' = \{\langle C, G \rangle, \langle G, H \rangle, \langle H, I \rangle\}$ are two disjoint MQPPs.

366 **C.** In the reduction graph $\mathcal{R}(T_5)$ of Fig.6.c:

367
$$P_1 = \{\langle A, B \rangle, \langle B, C \rangle, \langle C, D \rangle\} \cup \{\langle D, E \rangle, \langle E, F \rangle, \langle F, I \rangle\}$$

368
$$P_2 = \{\langle A, B \rangle, \langle B, C \rangle\}$$

369
$$P_3 = \{\langle D, E \rangle, \langle E, F \rangle\}$$

370 3 MQPPs with the nesting relations: $P_2 \subset P_1, P_3 \subset P_1$.

371 Let us describe the general structure of a *nested MQPP* P^1 . Let us call
 372 P_1^1, \dots, P_N^1 the normal decomposition of P^1 . The motifs P_n^1 are all quasi-
 373 isomorphic with each other and their union entirely covers P^1 . Therefore,
 374 to study the structure of P^1 one only needs to study the structure of one of
 375 them, say P_l^1 . Now let us consider the list of MQPPs included in P_l^1 . This
 376 list may be empty if P^1 is not nested and the MQPP is said to be *simple*.
 377 Otherwise, P_l^1 contains itself a list of MQPPs denoted P_l^{1j} for $j = 1..J$. If
 378 one of the P_l^{1j} is nested this decomposition recursively continues and P_l^{1j}

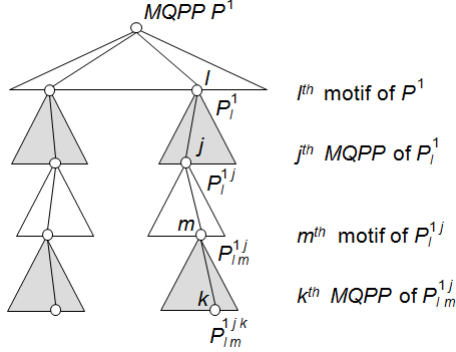


Figure 7: Indexing the component of a nested *MQPP*. Each motif is composed of a series of *MQPP*s and each *MQPP* is composed of a series of motifs.

379 can be decomposed into a list of quasi-isomorphic motifs denoted P_{lm}^{1j} for
 380 $m = 1..M$, and so on (see Fig.7). These series of decomposition forms a tree
 381 structure called the *MQPP tree* of P^1 . Interestingly, this tree contains many
 382 identical subtrees as indicated by the following property.

383 **Property 8** $\forall m \neq m', P_{l..m'}^{1j..k} \cong P_{l..m}^{1j..k}$.

384 **Definition 9 (Maximally Nested *MQPP*)** A maximally nested *MQPP*
 385 is a *MQPP* that is not strictly contained within a strictly greater *MQPP*.

386 **Definition 10 (Normal decomposition of a *DAG*)** The normal decom-
 387 position of a *DAG* D is the set \mathcal{N} of all its maximally nested *MQPP*s.

388 Therefore, the normal decomposition of a *DAG* contains only top level nested
 389 *MQPP*s which are either simple or decomposable in to finer *MQPP*s.

390 In general, the relative position of two *MQPP*s must respect constraints
 391 as indicated by the following property.

392 Let $P = \{e_1, \dots, e_n\}$ be a path. If $e_n = \langle x, y \rangle$ we say that P terminates
 393 through x .

394 **Property 9** Let P and P' be two intersecting *MQPP*s:

- 395 1. If P and P' starts from the same vertex then $P \subset P'$ or $P' \subset P$.
- 396 2. If P and P' are not nested then either:
 - 397 (a) at least one of the *MQPP*s terminates through the intersection
 - 398 point,

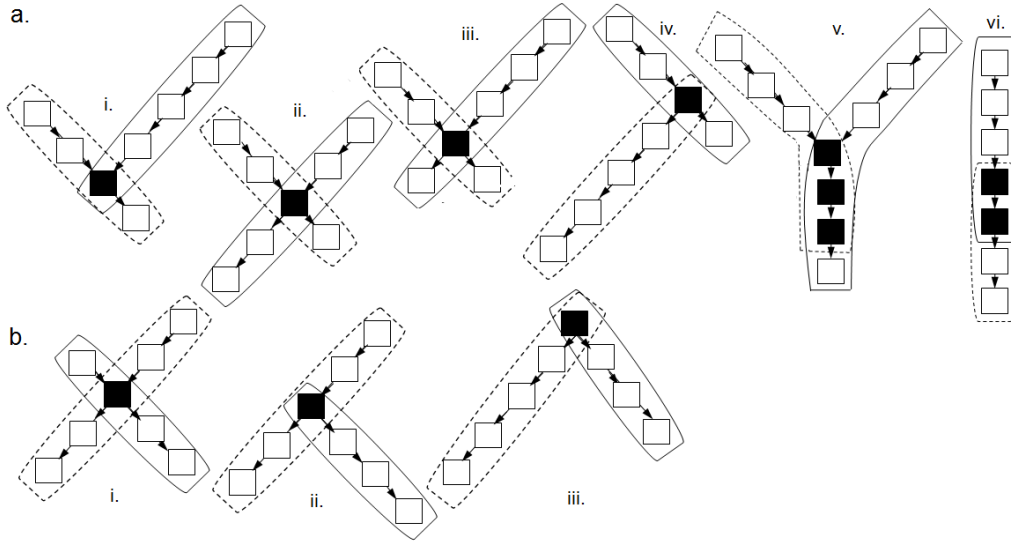


Figure 8: a. Patterns of relative positions of two intersecting *MQPPs*, b. Patterns of impossible situations of two intersecting *MQPPs*. See text for detail explanation

- 399 (b) *the intersection corresponds to a postfix of one of the two paths,*
 400 (c) *or both MQPPs belong to the same path.*

401 **Example 8** Let us illustrate the different situations of two non nested inter-
 402 secting *MQPPs* cited in Property 9. Fig.8.a shows some possible situations
 403 of two non nested intersecting *MQPPs*, and Fig.8.b some impossible ones,
 404 so in:

- 405 • Figs.8.a.i, 8.a.ii, 8.a.iii and 8.a.iv at least one of the *MQPPs* terminates
 406 through the intersection point,
- 407 • Fig.8.a.v the intersection corresponds to a postfix of one of the two
 408 paths,
- 409 • Fig.8.a.vi both *MQPPs* belong to the same path,
- 410 • Figs.8.b.i, 8.b.ii and 8.b.iii there is no *MQPP* which terminates through
 411 the intersection point.

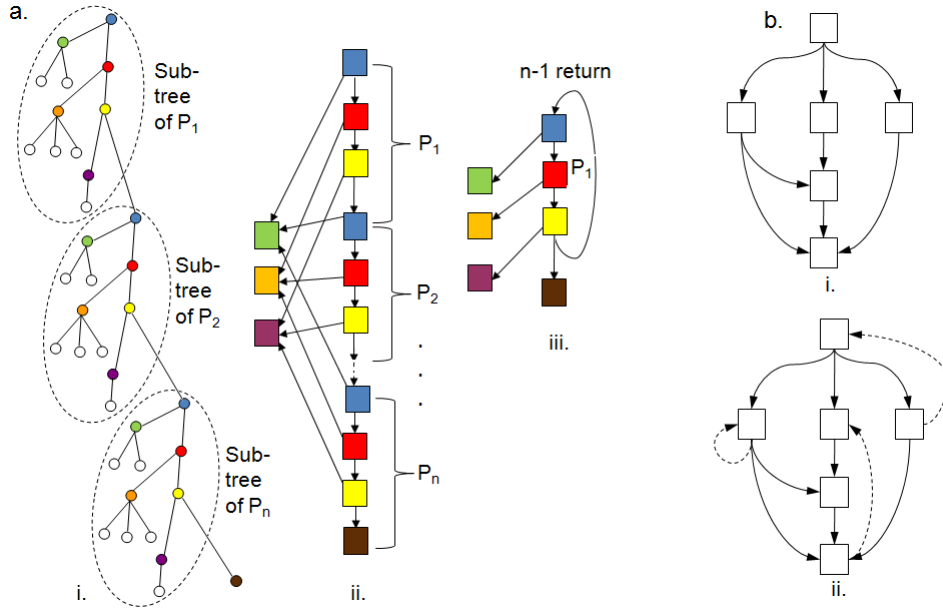


Figure 9: a.i. A height repeated branch, ii. the corresponding $MQPP = P_1 \cup \dots \cup P_n$ in the associated DAG , iii. the equivalent height reduced DAG including a return edge and a returns number $(n - 1)$, b.i. a DAG , ii. the original DAG augmented with return edges (dashed edges), the result is a DAG with return edges

412 5. Height compression of a reduction graph

413 Each $MQPP$ can be decomposed in a unique manner as a sequence of
 414 quasi-isomorphic paths (normal decomposition) P_1, \dots, P_n (as in Fig.9.a.ii).
 415 We will make use of this repeated pattern to compress these $MQPP$ s. This
 416 will be made by allowing introduction of loops in the original DAG s, as in
 417 Fig.9.a.iii. Let us introduce the notion of DAG with return edges.

418 5.1. DAG s with Return Edges

419 A DAG with return edges is constructed from a DAG by adding to it
 420 edges that induce oriented cycles, Fig.9.b.

421 **Definition 11 (DAG with return edges)** Let $D = (V, E)$ be a DAG .
 422 We consider a set E' of edges in $V \times V$ such that:

- 423 • $\forall (x, y) \in E', y \prec x$ in D .

- 424 • As for any edge of D , we associate a weight $n(x, y)$ with each edge
425 $(x, y) \in E'$.
- 426 • In addition we associate with each edge $(x, y) \in E'$ one of the children
427 z of x if any, such that $n(x, y) = n(x, z)$, z is called the exit vertex of
428 the cycle induced by the return edge (x, y) and denoted $ex(x, y)$.

429 The triplet $D' = (V, E, E')$ verifying these conditions is called DAG with
430 return edges, where E' is the set of return edges of D' . \mathcal{D}_\uparrow denotes the class
431 of all DAGs with return edges.

432 **Property 10** Let $B = (e_1, \dots, e_n)$ an elementary cycle in a DAG with return
433 edges $D = (V, E, E')$. There is a unique return edge $e_k \in E'$ in B denoted
434 $re(B)$. We called the path $\{e_{k+1}, \dots, e_n, e_1, \dots, e_{k-1}\}$ the principal path of B
435 denoted $pp(B)$, and the vertex $ex(e_k) = z$ the exit vertex of the cycle B ,
436 denoted $ex(B)$.

437 Furthermore, we say that $x \in pp(B)$, for each vertex x in $pp(B)$. In the
438 DAG with return edges of Fig.10.e, for the elementary cycle $B_2 = ((x, y),$
439 $(y, z), (z, w), (w, x))$: $re(B) = (w, x)$, $pp(B) = \{(x, y), (y, z), (z, w)\}$ and
440 $ex(B) = e$.

441 The MQPPs correspond to maximally repeated patterns on the paths of
442 a DAG. It is thus possible to compress the DAG in height by compressing
443 these repeated motifs. This will lead to construct DAGs with return edges
444 representing the compressed trees. As we will show, the loop in these graphs
445 can only be either nested or disjoint. This leads to introduce a special kind
446 of DAGs with return edges, namely DAGs with nested returns.

447 5.2. DAG with nested returns

448 **Definition 12 (DAG with nested returns)** We define a DAG with nest-
449 ed returns as a DAG with return edges in which any two elementary cycles B_1
450 and B_2 (with respective vertex sets V_1 and V_2) are either nested ($V_1 \cap V_2 = V_1$
451 or $V_1 \cap V_2 = V_2$) or disjoint ($V_1 \cap V_2 = \emptyset$).

452 We denote the class of all DAGs with nested returns by $\mathcal{D}_\uparrow\uparrow$. Note that
453 $\mathcal{D}_\uparrow\uparrow \subset \mathcal{D}_\uparrow$.

454 **Example 9** Fig.11.a shows examples of patterns of vertices of DAGs $\in \mathcal{D}_\uparrow\uparrow$
455 and in Fig.11.b DAGs $\notin \mathcal{D}_\uparrow\uparrow$.

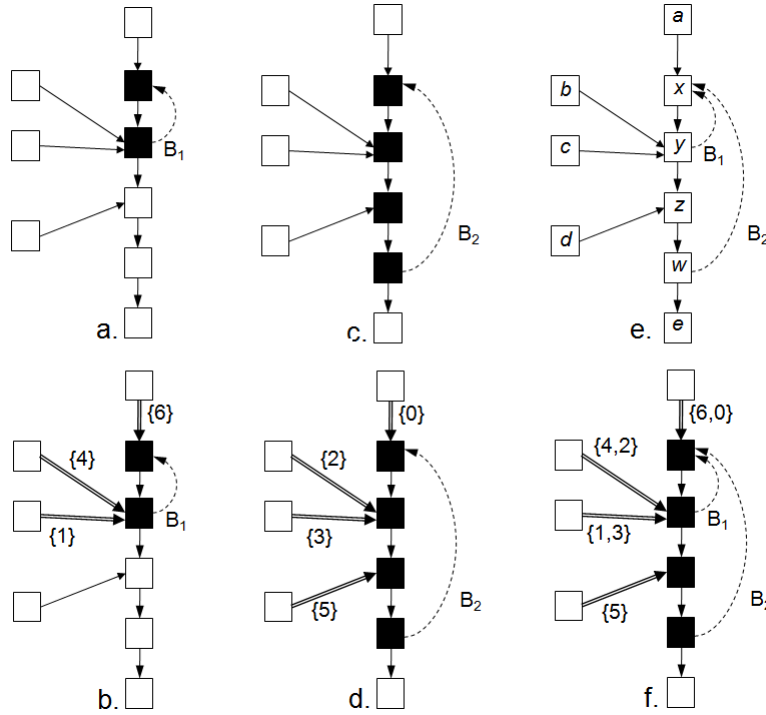


Figure 10: a. Cycle B_1 in a pattern of a DAG with returns D , b. The multiplicities of the set of entering edges in B_1 (Edges in bold), c. Cycle B_2 in the same graph D , d. The multiplicities of the set of entering edges in B_2 , e. the DAG D with the nested cycles B_1 and B_2 , f. the merged sets of multiplicities of entering edges in cycles B_1 and B_2

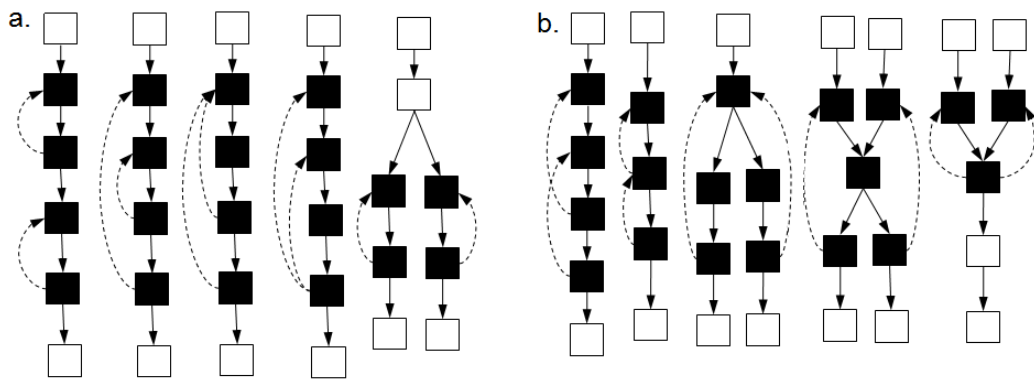


Figure 11: a. Patterns of valid DAGs in $\mathcal{D}_{\uparrow\uparrow}$, b. Patterns of DAGs not in $\mathcal{D}_{\uparrow\uparrow}$, (Dashed edges represent return edges)

456 Let $\mathcal{B}(D)$ denote the set of all cycles in a given $D \in \mathcal{D}_{\uparrow\uparrow}$. If $B_1, B_2 \in \mathcal{B}(D)$,
 457 the inclusion relation is denoted by $B_1 \subset B_2$.

458 **Definition 13 (Entering edges in a cycle)** Given $D = (V, E, E')$ in $\mathcal{D}_{\uparrow\uparrow}$.
 459 Let B a cycle in $\mathcal{B}(D)$. The set of entering edges in B , denoted $\mathcal{E}(B)$, is the
 460 set of all edges $(x, y) \in E$ such that $y \in pp(B)$ and $x \notin pp(B)$.

461 **Definition 14 (Multiplicity of a cycle for an edge)** Given $D \in \mathcal{D}_{\uparrow\uparrow}$ and
 462 let $B \in \mathcal{B}(D)$. With each edge e in $\mathcal{E}(B)$ we associate an integer μ_B called
 463 the multiplicity of B for e .

464 From Fig.10.b, μ_{B_1} for (a, x) is 4, while it is 6 for edge (b, y) .

465 The intuitive idea is to define on each edge entering a loop the number of
 466 times this loop must be scanned, and these is the multiplicity of the loop for
 467 this edge, Figs.10.b and 10.d. However, Fig.10.e shows a *DAG* with nested
 468 returns composed by the fusion of *DAGs* with return edges of Figs.10.a and
 469 10.c, where $B_1 \subset B_2$. Thus, the set of entering edges in B_1 is a subset of
 470 entering edges in B_2 , Figs.10.b and 10.d. So the multiplicity of each entering
 471 edge in simultaneously B_1 and B_2 correspond to the list of the multiplicities
 472 of B_1 and B_2 ordered according to the nesting order of the loops starting
 473 from the inner one, Fig.10.f. More details are given in Fig.12.a.

474 **Property 11** Given $D = (V, E, E')$ in $\mathcal{D}_{\uparrow\uparrow}$, $\forall x \in V$ there is at most an or-
 475 dered list of elementary nested cycles containing x , noted $\mathcal{B}(x)$ where $\mathcal{B}(x) =$
 476 $\{B_1, \dots, B_{b(x)}\}$, $b(x) = |\mathcal{B}(x)|$ and $B_1 \subset \dots \subset B_{b(x)}$.

477 In the *DAG* with nested returns of Fig.10.e there is two elementary cycles
 478 $B_1 = \{(x, y), (y, x)\}$ and $B_2 = \{(x, y), (y, z), (z, w), (w, x)\}$ with $B_1 \subset B_2$.
 479 Then $\mathcal{B}(x) = \mathcal{B}(y) = \{B_1, B_2\}$, $\mathcal{B}(z) = \mathcal{B}(w) = \{B_2\}$ and $\mathcal{B}(e) = \emptyset$.

480 Our aim is to show that *DAGs* with nested returns are in general com-
 481 pressed versions of *DAGs*, where the cycles can be unfolded to construct
 482 *DAGs*. Expanding a cycle in a *DAG* with nested returns corresponds to
 483 unfolding the return edge by successive repetitions of the principal path of
 484 the cycle. Let us illustrate this process on the graph of Fig.12.b from which,
 485 at first, the loop B_1 is expanded. Let $\mu_{max} = 2$ be the maximal multiplicity
 486 of all entering edges in B_1 . Expanding B_1 consists of:

- 487 • unfolding its return edge by the repetition of the principal path of B_1
 488 μ_{max} times. Let $P_1, \dots, P_{\mu_{max}+1}$ be the obtained repeated patterns with
 489 $P_1 = pp(B_1)$.

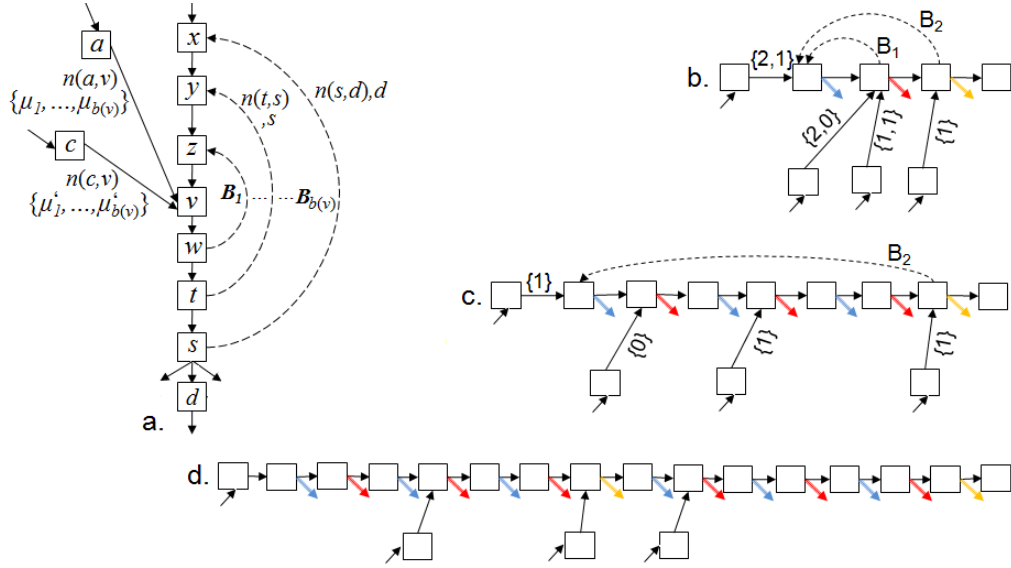


Figure 12: a. A general pattern of a formal detailed DAG with nested returns, b. A pattern of a labeled DAG with nested returns D , c. expansion of the inner cycle of D , d. completely expanded DAG . Vertices from which start thick edges of the same color have a common set of descendant vertices

- redirect each entering edge in B_1 , of multiplicity μ_{max} , into the equivalent entering edge in the new expanded path, i.e. edge of the same pattern in $P_{\mu_{max}-\mu+1}$.

Fig.12.c shows the resulting DAG with nested returns. The expansion of the remaining loop B_2 is performed in the same manner, Fig.12.d.

Property 12 Given $D = (V, E, E')$ in $\mathcal{D}_{\uparrow\uparrow}$, it is possible to unfold D into a DAG in time of $O(\mathcal{U} |V| |E'| deg^+(\mathcal{D}_{\uparrow\uparrow}))$.

\mathcal{U} is the maximal multiplicity of all cycles of $\mathcal{D}_{\uparrow\uparrow}$.

In the resulting DAG of Fig.12.d, we observe that the obtained expanded path forms a $MQPP$.

5.3. Compression of a DAG as a DAG with nested returns

We have just seen that $DAGs$ with nested returns can be unfolded into $DAGs$ by expending the cycles. We now consider the reciprocal question of compressing a DAG as a DAG with nested returns. To this end we will first

504 analyze how *MQPPs* can be folded and then apply the resulting algorithm
 505 to a maximal set of *MQPPs* in the original *DAG*.

506 5.3.1. Compression of a *MQPP*

507 Let P be a *MQPP* in a *DAG* D and P_1, \dots, P_n be its normal decomposi-
 508 tion. The idea is to replace the sequence P_1, \dots, P_n by a loop over P_1 repeated
 509 n times. In practice the original *DAG* must be edited in the following way.

510 **General compression algorithm of a *MQPP*:** Let $D = (V, E, E')$ be a
 511 *DAG* of $\mathcal{D}_{\uparrow\uparrow}$, and let $P = P_1 \cup \dots \cup P_n$ the *MQPP*(x, y) with $P_i = x_{i,1}P_i x_{i,m}$
 512 ($1 \leq i \leq n$, $m \geq 2$), $x = x_{1,1}$ and $y = x_{n,m}$, as represented in Fig.13.a. We
 513 denote by B_P the cycle obtained from the compression of P , and the resulting
 514 *DAG* with nested returns by $D_{/P}$. Then $D_{/P}$ is computed in the following
 515 steps:

516 1. Creation of a return edge, Fig.13.b:

- 517 • add a return edge to E' from $x_{1,m-1}$ to x ,
- 518 • this edge has a weight $\omega = n(x_{i,m-1}, x_{i,m})$,
- 519 • this edge is also augmented with the information that when the
 520 loop ends, scanning should resume on the exit vertex y of $x_{n,m-1}$,
 521 labeled by χ .

522 2. Redirection of entering edges of the *MQPP*, Fig.13.c:

- 523 • Replace each entering edge $(z, x_{i,j})$ of the *MQPP* by the equiva-
 524 lent entering edge in the cycle B_P , i.e. edge $(z, x_{1,j})$,
- 525 • this edge has a weight $n(z, x_{i,j})$,
- 526 • its multiplicity list corresponds to the multiplicity list of the edge
 527 $(z, x_{i,j})$ augmented by the multiplicity of the loop B_P for this edge,
 528 that is $n - i$.

529 3. Suppression of the *MQPP* repeated patterns, Fig.13.d:

- 530 • Remove edges which belong to the paths P_2, \dots, P_n ,
- 531 • the last edge of P_1 ($x_{1,m-1}, x_{1,m}$) is redirected on the exit vertex
 532 χ . It is thus replaced by the edge $(x_{1,m-1}, \chi)$ whose weight is also
 533 ω ,
- 534 • Edit the sets V , E and E' .

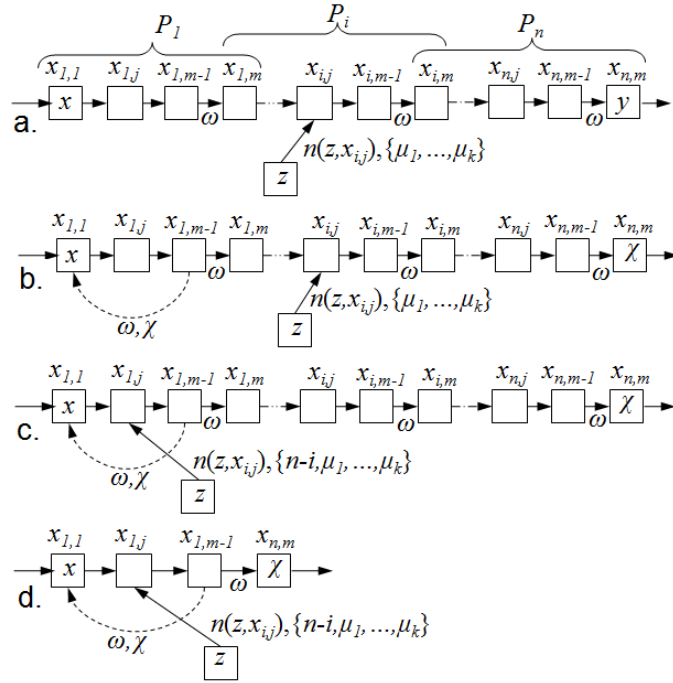


Figure 13: a. Formal pattern of a MQPP $P = P_1 \cup \dots \cup P_n$, b. creation of the return edge of P , c. redirection of an entering edge of P into its pattern in P_1 , d. Suppression of the repeated patterns P_2, \dots, P_n

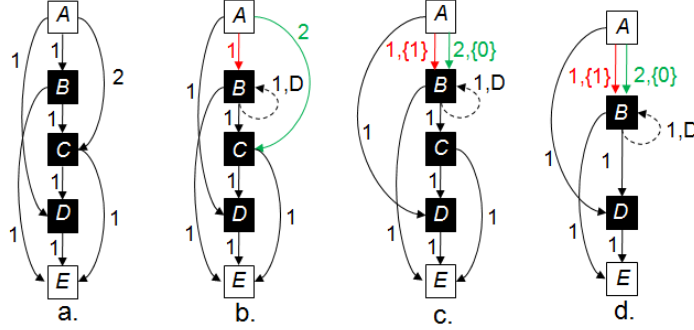


Figure 14: a. DAG $\mathcal{R}(T_1)$ b. creation of the return edge, c. redirection of the entering edges of P (colored edges), d. $\mathcal{R}(T_1)_P$

535 **Property 13** *The algorithm can be applied on the MQPP in time $O(|E|)$.*

536 **Example 10** Let us compress the MQPP $P = P_1 \cup P_2$ with $P_1 = \langle B, C \rangle$
 537 and $P_2 = \langle C, D \rangle$ of Fig.2.b. The compressed graph $\mathcal{R}(T_1)_P$ of Fig.14.c is
 538 computed by the following steps:

- 539 1. create the return edge (Fig.14.a):
 540 $re(B_P) = (B, B)$ of weight 1 and exit vertex D ,
- 541 2. redirect all entering edges of P into their patterns in B_{P_1} (Fig.14.b):
 542 edge $((A, B), 1, \{\})$ is replaced by $((A, B), 1, \{1\})$,
 543 edge $((A, C), 2, \{\})$ is replaced by $((A, B), 2, \{0\})$,
- 544 3. suppression of C and all its incident edges, and redirection of edge
 545 (B, C) on the exit vertex D (Fig.14.c).

546 Note on this example that due to the MQPP compression several edges can
 547 appear between two vertices.

548 5.3.2. Gain of MQPP compression

549 Let us consider a MQPP P in $D \in \mathcal{D}_{\uparrow\uparrow}$, with $P = P_1 \cup \dots \cup P_n$, and
 550 $P_i = x_{i,1}P_i x_{i,m}$ ($1 \leq i \leq n$, $m \geq 2$). Let us denote $|\overline{V}_P|$ and $|\overline{E}_P|$ the
 551 number of vertices and respectively the number of edges removed from P
 552 compression. Formally

$$|\overline{V}_P| = |V_P| - |V_{P_1}|, |\overline{E}_P| = \left(\sum_{i=2}^n \left(\sum_{j=1}^{m-1} deg^+(x_{i,j}) \right) \right) - 1.$$

553 Let α be the size of the computational representation of a vertex in V , and
 554 let β be the size of the computational representation of an edge in E .

555 **Definition 15 (compression gain of a MQPP)** We define the P com-
 556 pression gain, noted $g(P)$, by $g(P) = \alpha|\overline{V_P}| + \beta|\overline{E_P}|$.

557 **Property 14 (compression gain of two MQPPs)** Given two paths
 558 $P, P' \in MQPP(D)$. If P and P' are either disjoint or nested, then the gain

$$g(\{P, P'\}) = g(P) + g(P').$$

559 Let P and P' two MQPPs in a DAG D . $D_{/P/P'}$ denotes the DAG with
 560 nested returns obtained by successive compressions of P and P' , and the gain
 561 $g(D_{/P/P'}) = g(\{P, P'\})$.

562 *5.3.3. Selection of a maximal set of non intersecting MQPPs with maximal*
 563 *compression gain*

564 If two MQPPs are intersecting but no nested then the compression of one
 565 of them may impair or prevent the compression of the other. Therefore as
 566 the compression of the two MQPPs is mutually exclusive, the compression
 567 of the DAG will depend on the choice of the compression of either of them.
 568 We say that the DAG compression is ambiguous.
 569 Given a DAG D , let M be a subset of $MQPP(D)$.

570 **Definition 16** M is unambiguous for compression if $\forall P, P' \in M$, P and P'
 571 are either nested or disjoint.

572 In other terms, M is unambiguous for compression if all its MQPPs can
 573 be compressed independently. Therefore, if M is ambiguous for compres-
 574 sion one needs to make choices to eliminate ambiguity in compression while
 575 maximizing the compression gain.

576 For this, consider the equivalence relation \wedge on M such that:

$$577 \quad P \wedge P' \Leftrightarrow P \text{ and } P' \text{ are neither disjoint or nested.}$$

578 Let $\{M_i\}_i$ be the set of equivalence classes of \wedge (of Fig.15.a). In a class M_i
 579 MQPP's are pairwise intersecting with MQPP's of the same class. Note
 580 that classes themselves are disjoint.

581 Our problem is thus to select a subset of MQPPs in M_i such that this
 582 subset is unambiguous for compression, and the total gain of these MQPP's
 583 is maximal, Fig 15.b.

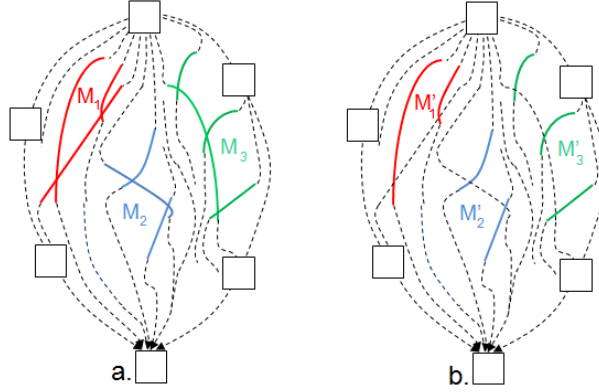


Figure 15: a. Pattern of a DAG with M_1, M_2, M_3 the equivalence classes of \wedge , i.e. maximal $MQPP$ sets ambiguous for compression (paths of the same color belongs to the same set M_i), b. the corresponding unambiguous subsets M'_1, M'_2, M'_3

584 To solve this problem, let us model the intersection map in class M_i as a
 585 graph: $G_{M_i} = (V_{M_i}, E_{M_i})$ such that each vertex of V_{M_i} represents a $MQPP$
 586 of M_i , and there is an edge between two vertices P and P' of V_{M_i} if P and P'
 587 are either disjoint or nested. With this graph definition we can characterize
 588 sets of $MQPP$ s unambiguous for compression:

589 **Property 15** M_i is unambiguous for compression if and only if G_{M_i} is a
 590 complete graph.

591 Therefore, our problem is reduced to finding in the graph G_{M_i} a clique
 592 $G_{M'_i}$ (i.e. a complete subgraph of G_{M_i}) with maximal gain. This optimization
 593 problem is known to be NP-complete (Feige et al. (1991)), and several ap-
 594 proaches have been developed to solve it (Battiti and Protasi (2001); Bomze
 595 et al. (1999); Feige (2004)). Here we use the *TABU algorithm* (Gendreau
 596 et al. (1993)), as described in section 15 of the Online supplementary ma-
 597 terial. Given a class M_i the algorithm returns a maximal subset M'_i unam-
 598 biguous for compression with maximal gain.

599 For a DAG D , let $M = \cup_i M_i$ and $M' = \cup_i M'_i$, and let us denote by
 600 $\overline{MQPP}(D)$ the maximal set of $MQPP$ s of D unambiguous for compression,
 601 with:

$$\overline{MQPP}(D) = MQPP(D) \setminus (M \setminus M')$$

602 Where \setminus denotes the sets difference.

603 Let $\mathcal{N}(D)$ be the normal decomposition of a DAG D , $\overline{MQPP}(D)$ allows us
 604 to eliminate the compression ambiguity on the full DAG D .

605 **Definition 17 (Unambiguous normal decomposition of a DAG)**

606 *The unambiguous normal decomposition of D , \mathcal{N}^* , is the set of all its maxi-*
 607 *mally nested MQPPs unambiguous for compression, i.e. the set*

$$\mathcal{N}^* = \mathcal{N}(D) \cap \overline{MQPP}(D).$$

608 Section 16 of the Online supplementary material provides examples of
 609 compressed graphs obtained from graphs of Fig.6.

610 *5.3.4. DAG compression evaluation:*

611 Given a DAG $D = (V, E)$. Let R be the corresponding DAG with nested
 612 returns with gain $g(R)$.

613 **Definition 18 (Compression factor of a DAG with nested returns)**

614 *The compression factor of a DAG with nested returns R is defined as:*

$$f(R) = \frac{g(R)}{\alpha|V| + \beta|E|}$$

615 where α, β are the size of the computational representations of respectively
 616 a vertex in V and of an edge in E . This measure represents the relative gain
 617 in size of the compressed graph compared to the original DAG.

618 *5.4. Algorithm for the height compression of a finite tree*

619 Based on the definition of MQPPs and on their compressibility property,
 620 it is possible to design an optimization algorithm to maximally compress a
 621 DAG in height.

622 **Property 16** *Let $D = (V, E)$ be a DAG, the compression in height, denoted*
 623 *$\mathcal{H}(D)$, can be computed in time $O(|E|^2 h(D) l(D))$.*

624 The main steps of the computation of $\mathcal{H}(D)$ are described as follows:

- 625 1. Compute the set of vertex signatures and then the set of edge signatures
 626 of D (see section 4.2).
- 627 2. Given the set of edge signatures compute the set $MQPP(D)$ (see sec-
 628 tion 4.3).

- 629 3. From $MQPP(D)$ determine the sets M then M' ,
630 and deduce $\overline{MQPP(D)}$ (see section 5.3.3).
631 4. Iterative computation of $\mathcal{H}(D)$:
632 (a) Initialization: Let $\mathcal{H} = (V, E, E')$ be the *DAG* with nested returns
633 for which the vertex and the edge set are equal to the vertex and
634 the edge set of D , and the set of return edges is an empty set.
635 (b) Repeat
636 i. compute $\mathcal{N}^*(\mathcal{H}) = \{P_1, \dots, P_n\}$ the unambiguous normal de-
637 composition of \mathcal{H}
638 ii. $\mathcal{H} = \mathcal{H}_{/P_1/\dots/P_n}$
639 until $\mathcal{N}^*(\mathcal{H}) = \emptyset$,
640 5. Return the *DAG* with nested returns $\mathcal{H}(D) = \mathcal{H}$ of compression factor
641 $f(\mathcal{H}(D)) = \frac{g(\mathcal{H}(D))}{\alpha|V| + \beta|E|}$.

642 Finally, if $\mathcal{R}(T)$ denotes the reduction of the tree T in width, if $\mathcal{H}(D)$
643 denotes the compression of a *DAG* D in height then the compression $\mathcal{R}^*(T)$ of
644 the tree T in *width and height* can be computed in time $O(|T|^2 \deg(T) h(T))$
645 such as:

$$\mathcal{R}^*(T) = \mathcal{H} \circ \mathcal{R}(T).$$

646 6. Application to the compression of plant structures

647 Most plants have a modular branching structure made up of the repetition
648 of basic modules such as leaves, stem portion between two nodes, shoots, etc.
649 (Bell (1991), Godin and Caraglio (1998)). These highly repetitive structures
650 are therefore well suited to assess our compression schemes. In this section,
651 we will investigate how different types of plant structures can give rise to
652 different types of compression in height.

653 6.1. Dichotomic Structures

654 Plants may build up dichotomic branching structures due to either the
655 subdivision of their apices or to sympodial branching (the main apex stop
656 to grow and two lateral apices resume the growth). Many plants show such
657 a type of organization including one of the initial form of the earth's plant
658 life, that is the *Cooksonia Caledonica* plant (Boyce (2008)) (Fig.16.a.i), and
659 the *Horneophyton lignieri* (Eggert (1974)) (Fig.16.a.ii).

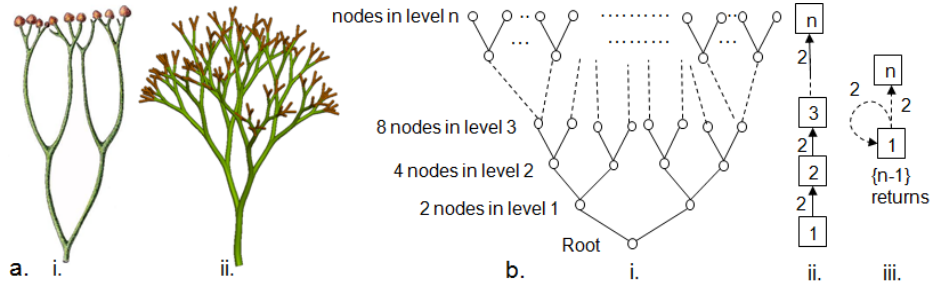


Figure 16: a. Examples of vegetal dichotomic structures: i. The *Cooksonia Caledonica* shape (Tandart et al. (2014)), ii. *Horneophyton* tree (Tandart et al. (2014)), b.i. a binary tree graph T , ii. the DAG D of its width reduction, iii. its height reduction graph $\mathcal{H}(D)$

660 Formally, the general pattern of a dichotomic structure may be repre-
 661 sented by a binary tree graph T (Fig.16.b.i). If n denotes the height of T
 662 then the DAG $D = \mathcal{R}(T)$ represents its width reduction (Fig.16.b.ii). Using
 663 our algorithm (see Property 16) this DAG can be further reduced in height
 664 in time $O(n^2)$ leading to a DAG with nested returns $\mathcal{H}(D)$ (Fig.16.b.iii) with
 665 a reduction factor of

$$666 \quad f(\mathcal{H}(D)) = 1 - \frac{2\alpha' + 2\beta'}{\alpha n + \beta(n-1)}$$

667 where α, β are the size of the computational representations of respectively a
 668 vertex and of an edge in a DAG D , and α', β' are the size of the computational
 669 representations of respectively a vertex and of an edge in $\mathcal{H}(D)$. These
 670 notations will be used all through this section.

671 6.2. Structures with multi-scale periodicity

672 Fish bone-like structures present a typical example of structure of multi-
 673 scale periodicity. They can be noticed, for example, in axial trees whose
 674 growth is *monopodial* (i.e. trees having a growing main stem that keeps on
 675 growing while producing lateral branches). Such a structure is thus composed
 676 of a principal axis on which several elements are repeated (branches, leaves,
 677 fruits,...). The *Date Palm* illustrates the wide variety of plants or plant parts
 678 showing a fish bone-like structure (Fig.17.a.i). It can be abstracted as the
 679 graph T represented in Fig.17.a.ii, where the main axis represents the trunk of
 680 the palm tree and the lateral branches represents the composed leaves. This
 681 structure can be compressed in width leading to the graph D of Fig.17.a.iii
 682 that can itself be further compressed in height as $\mathcal{H}(D)$ (Fig.17.a.iv) with
 683 height compression factor of

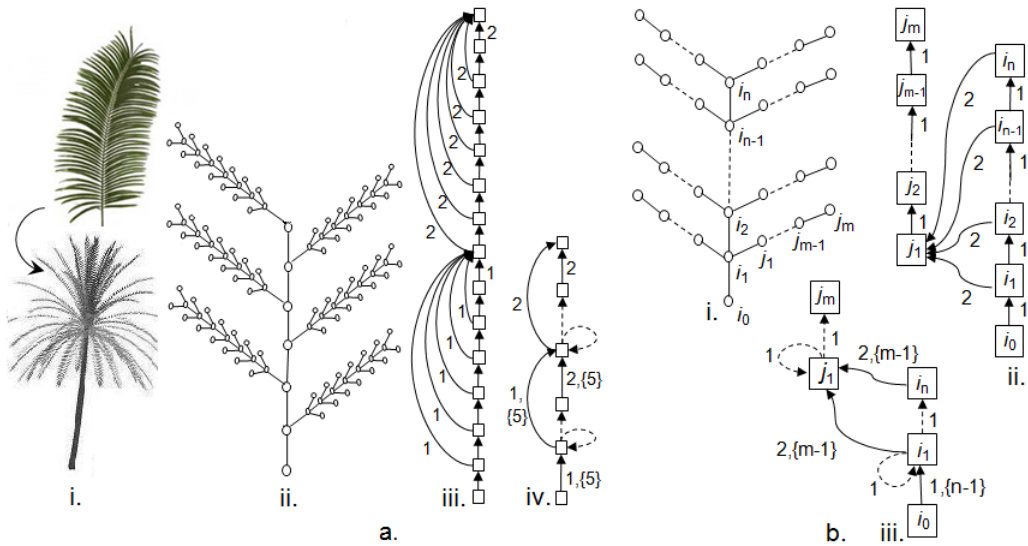


Figure 17: a.i. A model of the date palm (*Phoenix Dactylifera*) (Rhouma (1994)) and its lateral leaf modeled using the L-studio/Vlab (Prusinkiewicz et al. (2000); Abelson and DiSessa (1981)), ii. graphical representation T of the date palm. iii. the corresponding DAG D , iv. its height reduced graph $\mathcal{H}(D)$, b.i. Pattern of a fish bone-like structure T , ii. its width reduction DAG D , iii. its height reduced DAG $\mathcal{H}(D)$

684

$$f(\mathcal{H}(D)) = 1 - \frac{6\alpha' + 9\beta'}{\alpha(n+m+2) + \beta(2(n+m)-1)}$$

685

where $2n$ is the number of leaves on each lateral branch, and m the number of all lateral branches.

686

687

More generally, a fish bone-like structure can be compressed in height both on the lateral branches (j_1, \dots, j_m) and on the main axis (i_1, \dots, i_n), Fig.17.b.i. The width reduction of T produces the DAG D (Fig.17.b.ii).

688

689

Then the height reduction $\mathcal{H}(D)$ of D (Fig.17.b.iii) is computed in time $O((n+m)^3)$ with a height compression factor

690

691

$$f(\mathcal{H}(D)) = 1 - \frac{5\alpha' + 7\beta'}{\alpha(n+m+1) + \beta(2n+m-1)}$$

692

Note the two cycles of $\mathcal{H}(D)$ express respectively the repetition of motifs on lateral branches and the repetition of motifs on the main axis.

693

694

6.3. Structures with nested periodicity

695

696

The structures introduced above correspond to structures with a multi-scale periodicity, which is characterized by two levels of repeated patterns. In more complex multi-scale structures there may be several types of repeated

697

698

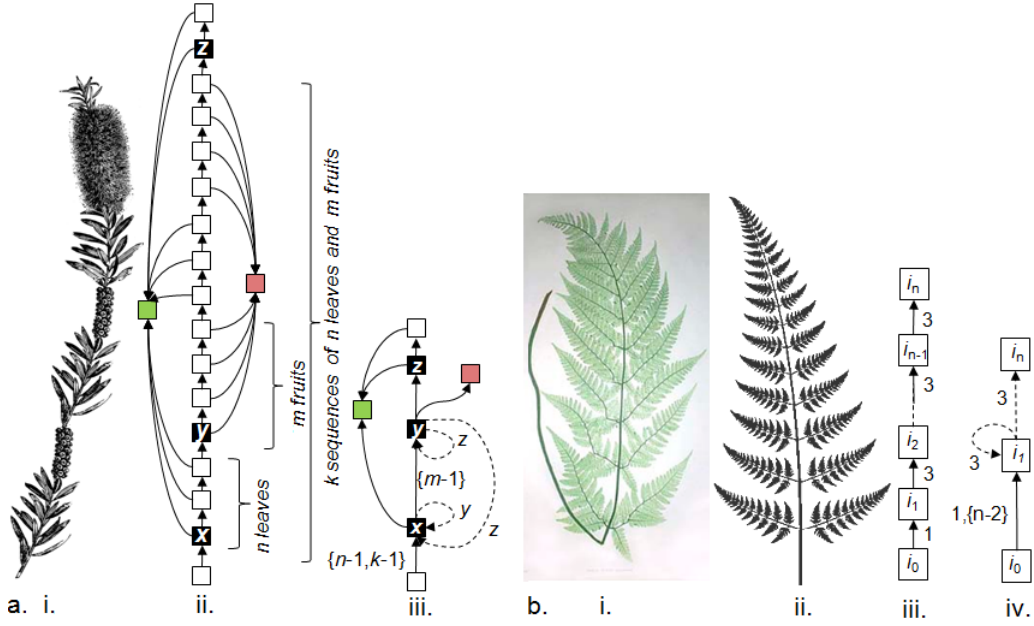


Figure 18: a.i. *Callistemon* branch (Rosser (2014); Nicholson (2014)), ii. Pattern of the corresponding DAG D , iii. its height reduced DAG $\mathcal{H}(D)$. Vertex in green represents a tree leaf, and pink vertex a fruit, b.i. A real fern of *Filicopsida Class* in the *Pteridophyta division* (Smith et al. (2006); Esculier and Ferreol (2014)), ii. Fractal fern illustrated using the OpenAlea Lab environment (Pradal et al. (2007)), iii. the associated DAG D , iv. its height reduced DAG $\mathcal{H}(D)$

699 patterns (leaves, flowers, twigs,...) that may alternate leading to nested peri-
700 odicity. The structure of the *Callistemon* branch (Stead and Butler (1983))
701 (Fig.18.a.i), in which groups of n lateral branches alternate with groups of
702 m fruits (k times), illustrates such a phenomenon. The DAG correspond-
703 ing to this structure can be represented in Fig.18.a.ii where the vertex in
704 green represents the class of tree leaves, and the pink vertex the class of the
705 fruits. The height reduction of the *Callistemon* branch $\mathcal{H}(D)$ (Fig.18.a.iii) is
706 computed in time $O(k^3(n+m)^3)$ with a height reduction factor

$$707 \quad f(\mathcal{H}(D)) = 1 - \frac{7\alpha' + 11\beta'}{\alpha(k(n+m)+3) + 2\beta(k(n+m)+2)}.$$

708 Note that the nested returns in $\mathcal{H}(D)$ express the nested periodicity,
709 where the outer loop corresponds to the repeated sequence (leaves, fruits),
710 and the inner loops represent respectively the leaves and the fruits repetitions.

711 6.4. Self-similar structures

712 Self-similarity is a conspicuous feature of many plants (Prusinkiewicz
 713 (2004)), that can be exploited to compress the plant structures. *Fern* pro-
 714 vides a typical example of such a self-similar structure (Fig.18.b.i). A model
 715 of this fern can be constructed in L-system (Prusinkiewicz et al. (1995)) -
 716 using a unique production rule (see section 17 of the Online supplementary
 717 material)- as represented in Fig.18.b.ii. The width reduction of a fern model
 718 of height n produces the DAG of Fig.18.b.iii, and the height reduction of D
 719 produces $\mathcal{H}(D)$ (Fig.18.b.iv) in time $O(n^3)$, with a height reduction factor

720
$$f(\mathcal{H}(D)) = 1 - \frac{3\alpha' + 3\beta'}{\alpha(n+1) + \beta n}.$$

721 For $n = 30$ the reduction factor is close to 93% showing the remarkable
 722 height compressibility of self-similar structures.

723 A general model of self-similar branching structures has been proposed
 724 by Godin and Ferraro (2010), called self-nested trees. To study and assess
 725 the quality of width compression algorithm the authors defined a database
 726 of self-nested branching structures made up by four families of purely self-
 727 nested trees (Fig.19.a and .b) together with versions of this trees altered with
 728 different levels of noise, Fig.19.c. In each family M_i ($i = 0..3$), starting from
 729 a self-nested template plant T_i , several noisy versions were created by varying
 730 the lateral branching probability (Fig.19.c).

731 Width and height compression were then applied on each tree T_i and all
 732 there noisy versions. Fig.20.a illustrates the width reduction graphs of some
 733 specimens of the M_3 family, and for which the height compression provides
 734 the graphs of Fig.20.b. For the four families, the obtained height compression
 735 factors depending on the branching probability (for α and β fixed to one unit)
 736 is shown in Fig.21.

737 One can notice that the *monopodial* template plants (T_0 , T_1 and T_2)
 738 are weakly height compressible structures, whereas the *sympodial* tree T_3 is
 739 remarkably height compressible. However as soon as noise is introduced in
 740 this tree, its height compressibility becomes weaker.

741 Overall, the height compressibility of the four families is low for small
 742 noise factor (probability ≥ 0.6). Nevertheless for increasing noise (probability
 743 < 0.6) height compressibility increases as the tree tends to become a linear
 744 structure.

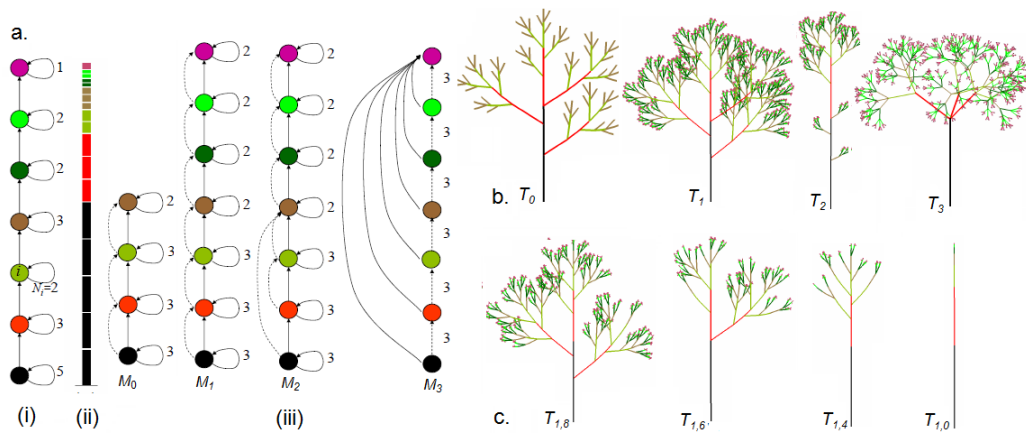


Figure 19: a. (i) The differentiation graph of a non-branching plant structure. (ii) The resulting axis structure, where component colors correspond to the differentiation graph states in which these components were created. The numbers attached to each loop indicate the number of steps a meristem stays in the corresponding state. (iii) Differentiation graphs used for the definition of the theoretical plants T_i ($i = 0..3$). Solid arrows correspond to possible transitions of the apical meristem states. Dashed arrows correspond to possible transitions from the apical meristem state to the axillary meristem states. b. the self-nested template plants T_i ($i = 0..3$), c. the noisy versions of the template plant T_1 with probability 0.8, 0.6, 0.4 and 0.0

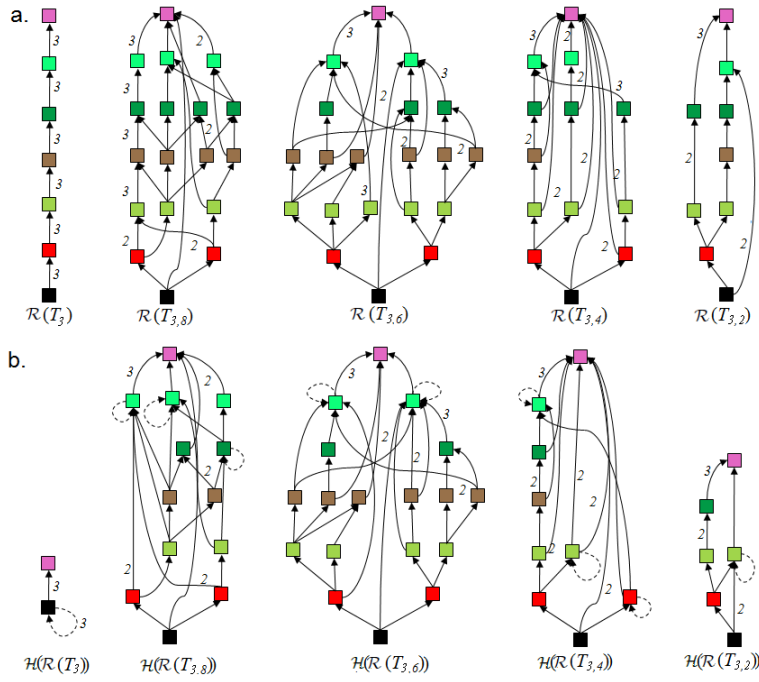


Figure 20: a. The width reduction graphs of the template plant T_3 and its noisy versions $T_{3,8}$, $T_{3,6}$, $T_{3,4}$ and $T_{3,2}$ ($T_{3,i}$ is the noisy tree of branching probability $0, i$). b. the corresponding height reduced graphs

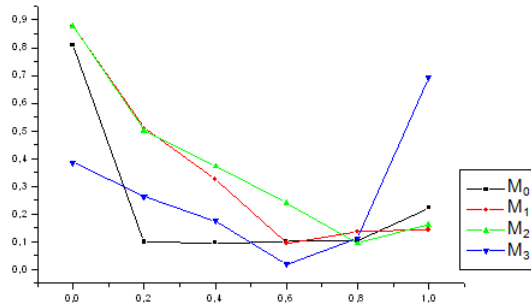


Figure 21: Height compression factor $f(\mathcal{H}(\mathcal{R}(T_i)))$ depending on the branching probability

745 **7. Conclusion**

746 This paper has considered the problem of height reduction from tree struc-
747 tures. This study was designed to complement the work previously done in
748 the context of lossless compression of unordered trees (Godin and Ferraro
749 (2010)), where the trees were compressed in width and not in height. Us-
750 ing the property that some tree structures exhibit height regularities, we
751 proposed to represent differently these special structures, characterized by
752 a height repeated pattern. These repetitions are associated with quasi-
753 isomorphic sub-trees which are regularly organized along one main branch.

754 Starting from a reduction graph of a given tree, the height reduction consi-
755 sts of searching of all collections of regularly height repeated patterns. Each
756 repeated pattern is replaced by a unique instance of this pattern augmented
757 with a loop in the compressed tree which is called a Reduction Graph with
758 Nested Returns. The efficiency of the compression algorithm is quantified by
759 a compression factor reflecting the ratio between the size of the *DAG* with
760 nested returns and the size of the initial *DAG*.

761 The method was then illustrated on different type of plant structures
762 and showed that in many cases height compression can add a significant
763 compression factor to the width compression, in particular on self-similar
764 plants.

765 **Acknowledgments**

766 The authors are grateful to the University Djillali Liabes of Sidi-Bel-
767 Abbes (Algeria) and to the INRIA-CIRAD-INRA Virtual Plants project-
768 team of Montpellier (France) who funded this research.

769 **Bibliography**

- 770 Abelson, H., DiSessa, A., nov 1981. Turtle geometry: The computer as a
771 medium for exploring mathematics. MIT Press, Cambridge, MA 39 (11).
- 772 Arber, A., 1950. Natural philosophy of plant form. University Press, Cam-
773 bridge, 210–217.
- 774 Barnsley, M. F., 1988. Fractals Everywhere. Academic Press, Dublin. (2nd
775 Edition, Morgan Kaufmann 1993; 3rd Edition, Dover Publications, 2012).
- 776 Barnsley, M. F., 2006. SuperFractals. Cambridge University Press.

- 777 Barthelemy, D., 1991. Levels of organization and repetition phenomena in
778 seed plants. *Acta Biotheoretica*, vol. 39, pp. 309-323.
- 779 Battiti, R., Protasi, M., 2001. Reactive local search for the maximum clique
780 problem. *Algorithmica* 29 (4): 610-637.
- 781 Bell, A., 1991. *Plant form: An illustrated guide to flowering plants*. Oxford
782 Univ. Press, Oxford.
- 783 Ben-Naoum, F., 2009. A survey on l-system inference. *Infocomp journal of*
784 *Computer Science*, Vol 8, No 3.
- 785 Bomze, I., Budinich, M., Pardalos, P., Pelillo, M., 1999. The maximum clique
786 problem. *Handbook of Combinatorial Optimization 4*, Kluwer Academic
787 Publishers, pp. 1-74.
- 788 Borchert, R., Honda, H., 1984. Control of development in the bifurcat-
789 ing branch system of *tabebuia rosea*: a computer simulation. *Botanical*
790 *Gazette*, Vol. 145, No. 2.
- 791 Boyce, C., 2008. How green was *cooksonia*? the importance of size in un-
792 derstanding the early evolution of physiology in the vascular plant lineage.
793 *Paleobiology* 34 (2): 179-194.
- 794 Deussen, O., Lintermann, B., 2002. *Digital Design of Nature: Computer*
795 *generated plants and organics*. ISBN 3-540-43608-5, Springer Verlag Berlin
796 Heidelberg.
- 797 Durand, J., Caraglio, Y., Heuret, P., Nicolini, E., 2007. Segmentation-based
798 approaches for characterising plant architecture and assessing its plasticity
799 at different scales. Poster, FSPM07, New-ZELAND, 4-9 November.
- 800 Durand, J., Guedon, Y., Caraglio, Y., Costes, E., 2005. Analysis of the plant
801 architecture via tree-structured statistical models: the hidden markov tree
802 models. *New Phytol*, vol. 166, no. 3, pp. 813-825.
- 803 Eggert, D., 1974. The sporangium of *horneophyton lignieri* (rhyniophytina).
804 *American Journal of Botany*, Vol. 61, No. 4, pp. 405-413.
- 805 Esculier, A., Ferreol, R., 2014. *Barnsley's fern.*,
806 <http://www.mathcurve.com/fractals/fougere/fougere.shtml>.

- 807 Feige, U., 2004. Approximating maximum clique by removing subgraphs.
808 SIAM Journal on Discrete Mathematics 18 (2): 219-225.
- 809 Feige, U., Goldwasser, S., Lovasz, L., Safra, S., Szegedy, M., 1991. Approxi-
810 mating clique is almost np-complete. Proc. 32nd IEEE Symp. on Founda-
811 tions of Computer Science, pp. 2-12.
- 812 Gendreau, M., Soriano, P., Salvail, L., 1993. Solving the maximum clique
813 problem using a tabu search approach. Annals of Operations Research -
814 Special issue on Tabu search Volume 41 Issue 1-4, Pages 385-403.
- 815 Godin, C., Caraglio, Y., 1998. A multiscale model of plant topological struc-
816 tures. Journal of Theoretical Biology, Volume 191, Issue 1, Pages 1-46.
- 817 Godin, C., Ferraro, P., 2010. Quantifying the degree of self-nestedness of
818 trees: Application to the structural analysis of plants. IEEE/ACM Trans-
819 actions on Computational Biology and Bioinformatics, vol. 7, no. 4, pp.
820 688-703.
- 821 Halle, F., Oldeman, R., Tomlinson, P., 1978. Tropical trees and forests. An
822 architectural analysis. New-York: Springer-Verlag.
- 823 Harper, J., Rosen, B., White, J., 1986. The growth and form of modular
824 organisms. London, UK: The Royal Society.
- 825 Honda, H., 1971. Description of the form of trees by the parameters of the
826 tree-like body: Effects of the branching angle and the branch length on
827 the shape of the tree-like body. Journal of Theoretical Biology, Volume 31,
828 Issue 2, Pages 331-334, IN1-IN6, 335-338.
- 829 Lindenmayer, A., 1971. Developmental systems without cellular interactions,
830 their languages and grammars. Journal of Theoretical Biology, Volume 30,
831 Issue 3, pp. 455-484.
- 832 Lindenmayer, A., 1975. Developmental algorithms for multicellular organ-
833 isms: A survey of l-systems. Journal of Theoretical Biology, Volume 54,
834 Issue 1, Pages 3-22.
- 835 Lindenmayer, A., Rozenberg, G., 1972. Developmental systems and lan-
836 guages. in Proceeding STOC '72 Proceedings of the fourth annual ACM
837 symposium on Theory of computing, Pages 214-221.

- 838 Nicholson, G., 2014. The illustrated dictionary of gardening. div. vi. london,
839 england: L. upcott gill, http://etc.usf.edu/clipart/81800/81897/81897_
840 [callistemon.htm](http://etc.usf.edu/clipart/81800/81897/81897_callistemon.htm).
- 841 Pradal, C., Dufour-Kowalski, S., Boudon, F., Dones, N., 2007. The architec-
842 ture of openalea: A visual programming and component based software
843 for plant modeling. In Proceedings of the 5th International Workshop on
844 Functional Structural Plant Models, pp. 111-114.
- 845 Preparata, F., Yeh, R., 1973. Introduction to discrete structures for computer
846 science and engineering. Reading Menlo Park London: Addison-Wesley.
- 847 Prusinkiewicz, P., 1998. Modeling of spatial structure and development of
848 plants: a review. *Scientia Horticulturae* 74: 113-149.
- 849 Prusinkiewicz, P., 2004. Selfsimilarity in plants : integrating mathematical
850 and biological perspectives. in *Thinking in Patterns: Fractals and Related*
851 *Phenomena in Nature*, M. M. Novak, Ed. Singapore: World Scientific, pp.
852 103-118.
- 853 Prusinkiewicz, P., Hammel, M., Mech, R., Hanan, J., 1995. The artificial life
854 of plants. volume 7 of SIGGRAPH'95, Course Notes, pages 1-1
855 1-38, ACM Press.
- 856 Prusinkiewicz, P., Karwowski, R., Mech, R., Hanan, J., 2000. L-studio/cpfg:
857 a software system for modeling plants. In AGTIVE'99: Proceedings of the
858 International Workshop on Applications of Graph Transformations with
859 Industrial Relevance, 457-464.
- 860 Prusinkiewicz, P., Lindenmayer, A., 1990. The algorithmic beauty of plants.
861 New York: Springer Verlag.
- 862 Reffye, P. D., Edelin, C., Jaeger, M., 1989. La modelisation de la croissance
863 des plantes. *La Recherche*, 20 (207): 158-168.
- 864 Rhouma, A., 1994. Le palmier dattier en tunisie. I. Le patrimoine genetique,
865 Tunis, Arabesques, INRA Tunisie, GRIDAO France, PNUD/FAO, vol. 1,
866 254 p.
- 867 Rosser, C., 2014. The celia rosser medal for botanical art.,
868 [http://www.imis100ap1.com.au/FRBGM/FRBGM_Content/Botanic_](http://www.imis100ap1.com.au/FRBGM/FRBGM_Content/Botanic_Art_Folder/TABI/Celia.Rosser.Medal.aspx)
869 [Art_Folder/TABI/Celia.Rosser.Medal.aspx](http://www.imis100ap1.com.au/FRBGM/FRBGM_Content/Botanic_Art_Folder/TABI/Celia.Rosser.Medal.aspx).

- 870 Smith, A., Pryer, K., Schuettpelz, E., Korall, P., Schneider, H., Wolf, P.,
871 2006. A classification for extant ferns. *Taxon* 55(3): 705-731.
- 872 Stead, T., Butler, G., 1983. Your australian garden. No. 5 - Callistemons and
873 other Bottlebrushes D.G. Stead Memorial Wildlife Research Foundation.
- 874 Tandart, V., Gantet, P., Verger, A., Gantet, F., 2014. Les cormo-
875 phytes : Description et evolution des premieres plantes terrestres.,
876 <http://www.creaweb.fr/perso/bv/cormo2.html>.
- 877 Troll, W., 1937. Die wuchsform der baeume und straeucher. in *Vergleichende*
878 *Morphologie der hoeheren Pflanzen*, Ed. Berlin: Borntraeger, vol.1, pp.
879 636-643.
- 880 Viennot, X., Eyrolles, G., Janey, N., Arques, D., 1989. Combinatorial anal-
881 ysis of ramified patterns and computer imagery of trees. in *SIGGRAPH*
882 89: Proceedings of the 16th annual conference on Computer graphics and
883 interactive techniques. New York, NY, USA: ACM, pp. 31-40.