Exploration of a plant architecture database with the AMAPmod software illustrated on an apple tree hybrid family

Christophe Godin, Yann Guédon, Evelyne Costes

Agronomie 19 (1999) 163–184

Original article

Agriculture and Environment

Exploration of a plant architecture database with the AMAPmod software illustrated on an apple tree hybrid family

Christophe Godin[a]*, Yann Guédon[a], Evelyne Costes[b]

[a] Programme modélisation des plantes, Cirad, BP5035, 34032 Montpellier cedex 1, France
[b] Laboratoire d'arboriculture fruitière, Inra, 2, place Viala, 34060 Montpellier cedex 1, France

(Received 15 September 1998; accepted 9 February 1999)

Abstract – This paper describes the constitution and the statistical exploration of a plant architecture database using the AMAPmod system. In AMAPmod, plant architectures are represented by a formal model, called MTG, which is the central data structure of the system. A dedicated querying language AML enables the user to access plant architecture data with efficient built-in primitives. By combining these primitives, the user can extract various types of data that preserve a more or less important part of the structural information of the plant. Specific statistical tools have been designed to analyse data samples extracted from the plant architectures. Most of these tools, ranging from (hidden) Markovian models to dynamic programming comparison techniques, apply to samples of sequences. AMAPmod methodology is illustrated on an actual-scale example concerning a hybrid family of apple trees. (© Inra/Elsevier, Paris.)

AMAPmod software / plant architecture / database / statistical analysis of sequences / apple tree

Résumé – Exploration d'une base de données architecturales à l'aide du logiciel AMAPmod : application à une famille d'hybrides de pommiers. Ce papier décrit la constitution et l'analyse statistique d'une base de données contenant des descriptions d'architectures de plantes à l'aide du système AMAPmod. Dans AMAPmod, l'architecture des plantes est représentée par un modèle formel, nommé MTG, qui constitue la structure de données centrale du système. Un langage d'interrogation dédié, AML, permet à l'utilisateur d'accéder aux architectures des plantes à l'aide de primitives spécialisées et efficaces. En combinant ces primitives, des types de données variés permettant de préserver une part plus ou moins grande de l'information structurelle contenue dans la plante peuvent être extraits de la base de données. Des outils spécialisés ont été créés pour analyser les échantillons de données ainsi constitués. La plupart de ces outils, tant paramétriques (modèles markovien variés) que non paramétriques (algorithmes de comparaison de données fondés sur des techniques de programmation dynamique) s'appliquent à des séquences de données. La méthodologie AMAPmod

Communicated by Gérard Guyot (Avignon, France)

* Correspondence and reprints
godin@cirad.fr

est illustrée sur une base de données architecturale de taille réelle correspondant à une famille d'hybrides de pommiers.

logiciel AMAPmod / architecture des plantes / base de données / analyse statistique de séquences / pommier hybride

1. Introduction

During the 1970s, plant architecture progressively emerged as a new area of interest in different research domains, e.g. computer simulation [27], theoretical biology [14], botany [23, 24], agronomy [6], forestry [30], horticulture [29] and plant–environment interaction modelling [34]. All these domains considered plants from very different perspectives, but they all had in common a particular interest in the organisation of plant vegetative structures, i.e. in plant architecture.

In order to model plant architectures, people first attempted to measure plant organ parameters and other morphological characters. This was used to set the values of plant growth model parameters using empirical data [7, 11, 13, 28, 31], or to better understand the organisation of plant components, e.g.[2, 26, 33]. Recently, studies using plant architecture have entered a second phase, where the objective is to study variations of biological phenomena within crowns, e.g. [5, 39]. To achieve this goal, several works proposed to record topological information of plant components and to organise other information according to topology [12, 18, 25].

Godin and Caraglio recently introduced a plant representation formalism able to integrate several scales of description within a single model [16]. This formal representation is the central data structure of the AMAPmod system, which is a computational platform providing tools to create, explore and analyse plant architecture databases [17]. The representation model accounts for plant architectures measured at different scales (node, annual shoot and axis, for example), different dates and can integrate different types of attributes, geometrical or biological [17]. A set of dedicated statistical tools is used to identify remarkable structures or regularities in plant architecture which are not directly apparent in the data [18, 22]. Tools are also provided to compare biological structures, such as axes or branching systems, based on a comparison of their components [10, 21]. In agronomic applications, these tools have been used to characterise and compare genotype behaviour and cultural conditions [3, 4, 13, 21, 35, 39].

This paper describes the constitution of a plant architecture database and its exploration with the AMAPmod system. It emphasises how the different techniques and tools can be combined and applied using AMAPmod. Section 2 gives an overview of AMAPmod and outlines the different data structures and models that can be constructed and used in the system and gives an overall description of the system. The use of AMAPmod is then illustrated in section 3 on an actual-scale plant architecture database. The different steps in a typical exploration are successively illustrated using this database.

2. AMAPmod system overview

AMAPmod provides users with a methodology and corresponding tools to measure plants, create plant databases and analyse information extracted from these databases. This methodology can be depicted as shown in *figure 1*.

Plant architectures are described from field observations using a dedicated encoding language (*figure 1a, b*). These descriptions can then be decoded by the AMAPmod system which builds a specific internal representation of plant architecture (*figure 1c*). The resulting database can then be analysed with various statistical analysis tools. Plants can be graphically reconstructed and visualised in three dimensions. Various types of data

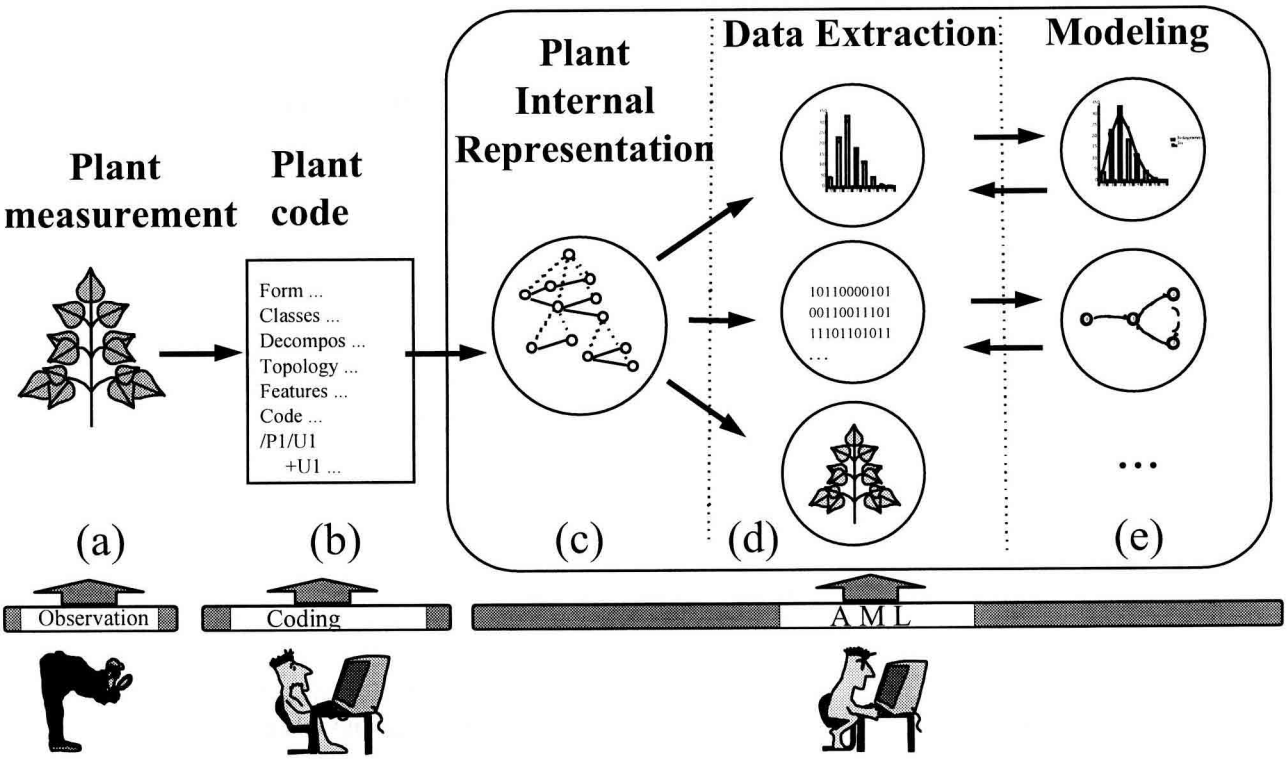


Figure 1. Synoptic of the AMAPmod system.

can be extracted and analysed from different viewpoints (*figure 1d*). Different families of probabilistic or stochastic models are provided in the system (*figure 1e*). These models are intended to be used as advanced statistical analysis tools for exploring in greater depth the information contained in the database. All these tools are available through a querying language called AML (AMAP modelling language) which enables the user to work on various objects, i.e. formal representation of plants, samples of data or models. AML provides the user with a homogeneous language-based interface to load, display, save, analyse or transform each type of object. Let us briefly review these AMAPmod components.

2.1. Plant architecture databases

Plants are formally represented in AMAPmod by multiscale tree graphs (MTGs)[16]. An MTG basically consists of a set of layered tree graphs, representing plant topology at different scales (internodes, growth units, axes, etc.). To build up MTGs from plants, plants are first broken down into plant components, organised into different scales (*figure 2a, b* p. 169). Components are given labels that specify their type (*figure 2b* p. 169, U = growth unit, F = flowering site, S = short shoot, I = internode). These labels are then used to encode the plant architecture into a textual form. The resulting coding file (*figure 2c* p. 169) can then be analysed by AMAPmod to build the corresponding MTG. Basically, in an MTG, the organisation of plant components at a given scale of detail is represented by a tree graph, where each component is represented by a vertex in the graph and edges represent the physical connections between them. At any given scale, the plant components are linked by relations which may be of two types. These types correspond to the two basic

mechanisms of plant growth, namely the apical growth and the branching processes. Apical growth is responsible for the creation of axes, by producing new components (corresponding to new portions of stem and leaves) on top of previous components. The connection between two components resulting from the apical growth is a 'precedes' relation and is denoted by a '<'. On the other hand, the branching process is responsible for the creation of axillary buds (these buds can then create axillary axes with their own apical growth). The connection between two components resulting from the branching process is a 'bears' relation and is denoted by a '+'. An MTG integrates – within a unique model – the different tree graph representations that correspond to the different scales at which the plant is described.

Various types of attribute can be attached to the plant components represented in the MTG, at any scale. Attributes may be geometrical (e.g. diameter of a stem, surface area of a leaf or three-dimensional (3D) positioning of a plant component) or morphological (e.g. number of flowers, nature of the associated leaf, type of axillary production – latent bud, short shoot or long shoot).

MTGs can be constructed from field observations using textual encoding of the plant architecture as described in [17] (*figure 2*). Alternatively, code files representing plant architectures can also be constructed from simulation programs that generate artificial plants [8]. The code files usually have a spreadsheet format and contain the description of plant topology in the first few columns and the description of attributes attached to plant components on subsequent columns.

2.2. The AMAPmod querying language: AML

AML is a functional language that provides users with an interactive interface to plant architecture databases. Databases, extracted data, statistical data samples and models are all represented in AML by dedicated data types that can be built and managed using specific functions. These functions are designed in order to provide users with high-level access to these generally complex objects. Let us first sketch the basics of the AML language.

From a practical viewpoint, AML is an interactive command interpreter that processes user's commands one after the other. AML's prompt `AML>` indicates that the system is waiting for user's input. After each input, the system evaluates the user's command and returns a message displaying the type of the object computed from the evaluation and its contents. For example, the command to read an array of integers from an ASCII file provides the following interaction:

```
AML> ARRAY("my_file_of_integers")

<ARRAY(INT)> [1,9,3,7,12,19]
```

In AML, all operations are expressed as function calls. AML contains a built-in set of functions, called primitives that can be split into several groups.

A first group provides a kernel of standard functions comprised of doing arithmetic, reading/writing data, storing variables, working on built-in data types, building new data types, displaying graphics, etc.

A second group of functions consists of primitives that enable the user to access the plant architecture database. This contains primitives for loading, exploring, extracting information, visualising and comparing MTGs.

A third group consists of statistical tools that can be used to explore and to analyse data extracted from MTGs. Primitives to estimate model parameters and to check for the validity of these models, to generate data samples from simulation are provided for each model.

Based on AML primitives, the user can build his own functions, which correspond to AML programs. A user-defined function is an expression containing one or several free variables that is given a name. For instance, consider the definition of a cone frustum with base diameter b, top diameter t and height h. The volume of a cone frustum can be defined in AML as a function, named cfvol, taking three arguments:

```
AML> cfvol(_b,_t,_h) = Pi * _h *
(_b^2 + _t^2 + _b*_t ) / 12
```

The underscore sign is used to make a distinction between free variables – used to define user functions – and normal computer variables, used to store values and computed objects. Free variables are not used to store any value, they only appear in expressions to identify a given term in different positions of an expression. Computer variables have a name not preceded by an underscore sign and appear in the left-hand side of an assignment expression:

```
AML> volume1 = cfvol(1,1,4)
               # volume1 is
               # a computer variable

<REAL> 3.14159 # value computed by
               # the call to cfvol
               # and stored in volume1
```

Several types of objects are defined in AML. These are first basic types such as INT, REAL, CHAR, STRING, DATE, BOOL, etc. These types can be combined to create new types using type constructors such as ARRAY, SET or LIST:

```
AML> array = [1,9,3] # creates an
# array containing 3 INTs

<ARRAY(INT)> [1,9,3]
```

ARRAYs are ordered sets of elements of identical types; SETs are sets of elements of identical types with no notion of order and containing no duplication of elements; LISTs are ordered sets of elements with possibly different types. Since ARRAY and LIST objects are ordered sets, their *i*th element can be defined:

```
AML> array@2 # extracts the second
# element from array

<INT> 9
```

All the preceding types correspond to objects that can be built by primitives that belong to the AML kernel. However, other AML primitives allow the user to build more specific objects. These specific objects are usually built by primitive constructors that have the name of the object type: for instance the primitive MTG can be applied to a file containing the code of a plant architecture to check its syntactic and semantic correctness and to build a MTG object containing the plant architecture information:

```
AML> my_plant = MTG("codefile")

<MTG> vtxnb = 1546  size = 10 kb
```

Similarly, an object of type HISTOGRAM can be built from an array of integers using the primitive Histogram:

```
AML> h1 = Histogram([1,2,2,3,3,2,4,
2,3,2])

<HISTOGRAM> sample size: 10  mean:
2.4 standard deviation: 0.843
```

or from a file containing the frequencies for each possible value.

Once specific objects such as MTG or HISTOGRAM are built, they can be displayed, plotted, transformed or compared using special primitives that apply to these objects. A sub-sample for instance can be extracted from the object h1 using the primitive ValueSelect by specifying the interval of values that must be kept:

```
AML> h2 = ValueSelect(h1,2,3)
          # selects a sub-sample of h1
          # [2,2,3,3,2,2,3,2]

<HISTOGRAM> sample size: 8  mean:
2.375 standard deviation: 0.518
```

Primitive functions have mandatory arguments and optional arguments. Mandatory arguments are identified according to their position in the argument list while optional arguments are given names and are not mandatory in the primitive function argument list. For example, in the primitive MTG to bound the number of errors output by the parser if any, the user may specify the optional variable MaxErrorNb as follows:

```
AML> my_plant = MTG("codefile",
MaxErrorNb -> 10)
```

In AML, loops can be carried out using iterators. The most common iterator enables the user to browse the elements of a set (either an ARRAY, a SET or a LIST) and to apply to each of them a particular function. For instance, the set of square val-

values corresponding to an array of integers can be computed as follows:

```
AML> square_vals = Foreach _x In
[1,2,3,4] : _x^2
<ARRAY(INT)> [1,4,9,16]
```

In section 3, we will illustrate several aspects of constitution and analysis of plant architecture databases by providing the AML queries that correspond to each step of a modelling session.

2.3. Types of extracted data

As mentioned above, various types of data can be extracted from MTGs. For each plant component in the database, attributes can be extracted or synthesised using the AML language. The wood volume of a component, for instance, can be synthesised from the diameter and the length of this component measured in the field. The type of measurement carried out in the context of architectural analysis emphasises the use of discrete variables which can be either symbolic, e.g. the type of axillary production at a given node (latent bud, short shoot or long shoot) or numeric (number of flowers in a branching structure). In general, a plant component can be qualified by a set of attributes, called a multivariate attribute. A plant component, for instance, could be described by a multivariate attribute made up of the volume, the number of leaves, the azimuth and the botanical type of the component.

Multivariate attributes correspond to the first category of data that can be extracted from MTGs. A second and more complex category of particular importance in AMAPmod is defined by sequences of – possibly multivariate – attributes. The aim of this category is to represent biological sequences that can be observed in the plant architecture. These sequences may have two origins: they can correspond to changes over time in the attributes attached to a given plant component. In this case, the sequences represent the trajectories of the components with respect to the considered attributes and the index parameter of the sequences is the observation date. Sequences can also correspond to paths in the tree topological structures contained in MTGs. In this case, the index parameter of the sequences is a spatial index that denotes the rank of the successive components in the considered paths. A spatially indexed sequence is a versatile data type for which the attributes of a component in the path can be either directly extracted or synthesised from the attributes of the borne components. In the latter case, all the information contained in the branching system can be efficiently summarised into a sequence of multivariate attributes, corresponding to the main axis of the branching system.

A third category of object can be extracted from MTGs, namely trees of – multivariate - attributes. Like sequences, these objects are intended to preserve part of the plant organisation in the extracted data. Tree structures represent the raw organisation of the components that compose branching structures of the plant at a certain scale of analysis.

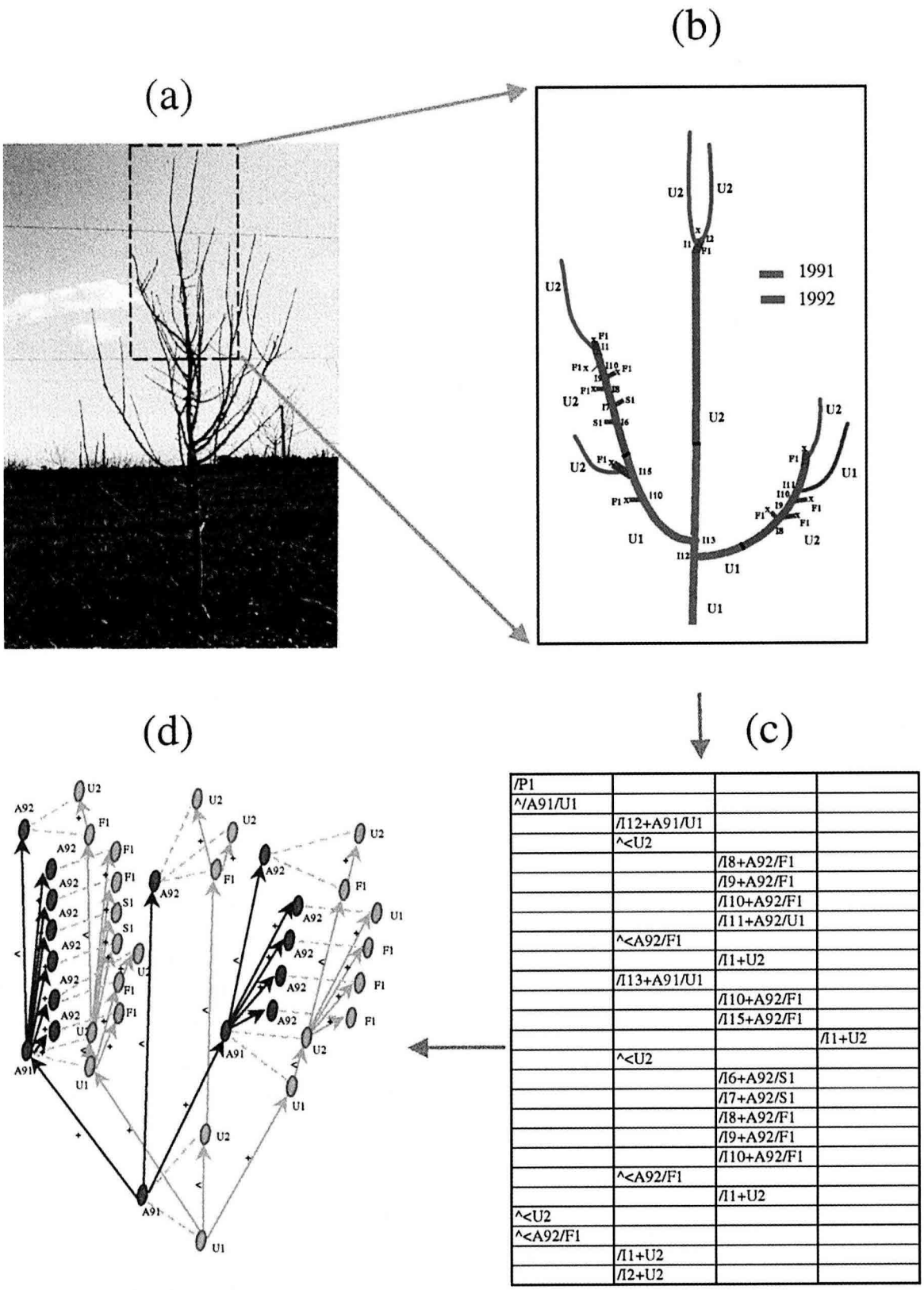
Data extracted from MTGs can thus be ordered according to their level of structural complexity: unstructured data, sequences, trees. These levels correspond to the different degrees to which the structural information contained in the MTG is summarised and are associated with different statistical analysis techniques.

2.4. Statistical exploration and model building

To explore plant architecture, users are frequently led to create data samples according to topological criteria on plant architecture. A wide range of AML primitives that apply to MTGs enable the user to express these topological criteria and select corresponding plant components. Samples of the three main structural data types can be created as described below.

2.4.1. Multivariate samples

Unstructured data samples can be created by computing the set of - possibly multivariate - attributes associated with a selected set of components, e.g. the number of flowers borne by components that appeared in the plant structure during 1995. Since a very large panoply of methods are



/P1			
^/A91/U1			
	/I12+A91/U1		
	^<U2		
		/I8+A92/F1	
		/I9+A92/F1	
		/I10+A92/F1	
		/I11+A92/U1	
	^<A92/F1		
		/I1+U2	
	/I13+A91/U1		
		/I10+A92/F1	
		/I15+A92/F1	
			/I1+U2
	^<U2		
		/I6+A92/S1	
		/I7+A92/S1	
		/I8+A92/F1	
		/I9+A92/F1	
		/I10+A92/F1	
	^<A92/F1		
		/I1+U2	
^<U2			
^<A92/F1			
	/I1+U2		
	/I2+U2		

Figure 2. Encoding plant architecture as an MTG. (a) A part of the plant is considered. (b) Plant components are identified within the crown. (c) The organisation of these components and their attributes are encoded in a code file. (d) An MTG representing the branching system can be built by AMAPmod from this code file. The plant representation at annual shoot scale is in red and at growth unit scale in yellow.

available in statistical packages for analysing multivariate samples, only a reduced core of tools have been integrated in AMAPmod for exploring these objects. If more specific statistical methods are required, the user can export data to other softwares such as SAS or S-PLUS.

2.4.2. Samples of multivariate sequences

The focus in AMAPmod is on data analysis tools for samples of sequences. In the context of plant architecture analysis, these objects present two advantages. On the one hand, part of the plant organisation is directly preserved in the sample through the notion of 'sequence' discussed above. On the other hand, the structural complexity of samples of sequences still remains tractable and efficient exploratory tools and statistical models can be designed for them [21, 22]. The AMAPmod system includes mainly classes of stochastic processes such as (hidden) Markov chains, (hidden) semi-Markov chains and renewal processes for the analysis of discrete-valued sequences. A set of exploratory tools dedicated to sequences built from numeric variables is also available, including sample (partial) autocorrelation functions and different types of linear filters (for instance symmetric smoothing filters to extract trends or residuals).

2.4.3. Samples of multivariate trees

The analysis of samples of tree structured data is a challenging problem. A sample of trees could represent a set of comparable branching systems considered at different locations in a plant or in several plants. Similarly, the development of a plant can be represented by a set of trees, representing different steps in time of a branching system. Plant organisation for this type of object is relatively well preserved in the raw data. However, this requires a higher degree of conceptual and algorithmic complexity. We are currently investigating methods for computing distances between trees which could be used as a basis for dedicated statistical tools.

AMAPmod contains a large set of tools for analysing these different types of samples, with special emphasis on tools dedicated to the analysis of samples of discrete-valued sequences. These tools fall into one of the three following categories:

- exploratory analysis relying on descriptive methods (graphical display, computation of characteristics such as sample autocorrelation functions, etc.);
- parametric model building;
- comparison techniques (between individual data).

The aim of building a model is to obtain adequate but parsimonious representation of samples of data. A parametric model may then serve as a basis for the interpretation of a biological phenomenon. The elementary loop in the iterative process of model building is usually broken down into the following three stages.

The specification stage consists in determining a family of candidate models on the basis of the results given by an exploratory analysis of the data and some biological knowledge.

The estimation stage consists in inferring the model parameters on the basis of the data sample. This model is chosen from within the family determined at the specification stage. Automatic methods of model selection are available for classes of models such as (hidden) Markov chains dedicated to the analysis of stationary discrete-valued sequences. In AMAPmod, the estimation is always made by algorithms based on the maximum likelihood criterion. Most of these algorithms are iterative optimisation schemes which can be considered as applications of the expectation-maximisation (EM) algorithm to different families of models [9, 19, 20]. The EM algorithm is a general-purpose algorithm for maximum likelihood estimation in a wide variety of situations best described as incomplete data problems.

The validation stage consists in checking the fit between the estimated model and the data to reveal inadequacies and thus modify the a priori specified family of models. In the AMAPmod system, theoretical characteristics can be computed from the estimated model parameters to fit the empirical characteristics extracted from the data and used in the exploratory analysis.

The parametric approach based on the process of model building is complemented by a non-parametric approach based on structured data alignment (either sequences or trees). Distance matrices built from the piece by piece alignments of a sample of structured data can be explored by clustering methods to reveal groups in the sample.

3. Illustration: exploring an apple tree orchard

Let us now illustrate the approach implemented in the AMAPmod system in a real application. To do this, we shall consider an apple tree orchard and show how a plant architecture database can be created from observations. Then, we shall use this database to illustrate the use of specific tools employed to explore plant architecture databases.

3.1. Biological context and data collection

The application is part of a general selection program conducted at Inra (*Institut national de la recherche agronomique*), and aims to improve apple tree species as regards morphological characters and more classical criteria such as fruit quality and disease resistance. In this particular example, two apple tree clones were chosen for their contrasted growth and branching habits. The first clone ('Wijcik') exhibits a very particular growth and branching habit, characterised by short internodes, great diameters and the absence of long axillary branches. By contrast, the second clone ('Baujade') exhibits many long and flexible branches. A population of 102 hybrids was obtained by crossing these two clones. The objective of this work was to study how morphological characters, such as the length of the internodes or the number of long lateral branches, are distributed within the progeny.

3.1.1. Creation of the database

The branching system borne by the 3-year-old annual shoot of the trunk is described for each individual. The branching system is first broken down into axes i.e. linear portions of stem derived from the same bud. Each axis is then divided into portions created during the same year (called annual shoots). When cessation and resumption of growth occur within a year, the annual shoot can be split into growth units, i.e. portions created over the same period (or between two resting periods). Finally, the growth units can be divided into internodes, i.e. portions of stem between two leaves. Regarding these successive decompositions, a given branching system is simultaneously considered at four scales. The different plant components and their connections are represented into a code file as explained previously.

In order to give a quantitative idea of the total resources necessary for an application of this size, it should be noted that all the measurements were carried out by a team of six persons over 5 days. The collected data, initially recorded on paper, were then computer-entered by one person over 20 days using a text editor, and consist of a file of approximately 16 000 lines of code. The corresponding MTG is constructed in 45 s on a SGI-INDY workstation. It contains about 65 000 components and some 15 000 attributes. The overall size of the database is 7 Mb.

3.2. 3D visualisation of real plants

To build the database associated with the collected data, the AMAPmod system is launched and an MTG is built from the encoded plant file:

```
AML> plant_database = MTG("apple-tree_code.txt")
```

The primitive `MTG` attempts to build a formal representation of the orchard, checking for syntactic and semantic correctness of the code file. If the file is not consistent, the procedure outputs a set of errors which must be corrected before applying a new syntactic analysis. Once the file is syntactically consistent, the MTG is built and is available in the variable `plant_database`. However, for efficiency reasons, the latest constructed MTG is said to be 'active': it will be considered as an implicit argument for most of the primitives dealing with MTGs. For example, to obtain the set of

vertices representing the plants contained in the database, i.e. vertices at scale 1, the primitive `VtxList` is used and applies by default to the active MTG `plant_database`:

```
AML>          plant_list          =
VtxList(Scale -> 1)
```

It is then possible to obtain an initial feedback on the collected data by displaying a 3D geometrical interpretation of a plant from the MTG. This notably allows the user to rapidly browse the overall database. For instance, a geometric interpretation of the 5th plant in the set of plants described in the MTG can be computed and plotted using the primitive PlantFrame as follows (*figure 3a*):

```
AML> geom_struct = PlantFrame
(plant_list@5)

AML> Plot(geom_struct)
```

Such reconstructions can be carried out even if no geometric information is available in the collected data. In this case, algorithms are used to infer the missing data where possible (otherwise, default information is used). In other cases, plants are precisely digitised and the algorithms can provide accurate 3D geometric reconstructions [5, 17, 36, 38].

Apart from giving a natural view of the plants contained in the database, these 3D reconstructions play another important role: they can be used as a support to graphically visualise how various sorts of information are distributed in the plant architecture. *Figure 3b* for example shows the organisation of plant components according to their branching order (trunk components have order 0, branch components have order 1, etc.). In AML, this would be obtained by the following commands:

```
AML>color_order(_x) =
     Switch Order(_x)
     Case 0: MediumGrey
     Case 1: DarkGrey
     Case 2: LightGrey Case 3: Black
     Default: White

AML> Plot(geom_struct, Color ->
color_order)
```

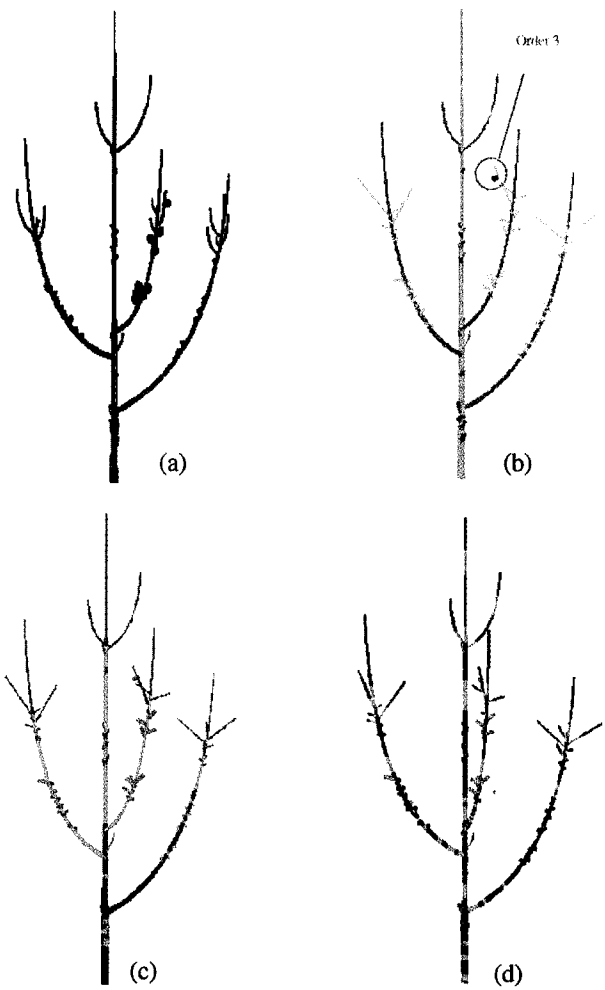


Figure 3. (a) 3D reconstruction of an apple tree recorded in the database. (b) Branching orders (green = 0, red = 1, light blue = 2, black = 3). (c) Years of growth (black = 1st year, green = 2nd year, red = 3rd year). (d) Growth rhythms (black = long internodes, red = scars, green = short internodes).

This representation emphasises different information related to the branching order: it can be seen in *figure 3b* that the maximum branching order is 3, that this order is reached only once in the tree crown, and that this occurs at a floral site (black component).

The use of the 3D representation of plant structure can also be illustrated in the context of plant growth analysis. The year in which each component grew can be retrieved from a careful analysis

of the plant morphological makers. If this information is recorded in the MTG, it is then possible to colour the different components accordingly. *Figure 3c* shows, for instance, that a branch appeared on the trunk during the first year of growth. This information can then be linked to other data, e.g. the branching order of a component or the number of fruits borne by a component, and thus provides deeper insight into the plant growth process.

Thanks to the multiscale nature of the plant representation, more or less detailed information can be projected onto the plant structure. Let us consider again the context of plant growth analysis. Plant growth is characterised by rhythms that result in the production of long internodes during periods of high activity and short internodes during rest periods (indicated on the plant by scares close together). This information, at the level of internodes, can be projected onto the plant 3D structure (*figure 3d*). Like the year of growth, this information enables us to access plant growth dynamics, but now, at an intra-year scale.

Finally, another use for the virtual reconstruction of measured plants is illustrated in *figure 4a, b*. These plants have been reconstructed from the MTG at the scale of each leafy internode. This enables us to obtain a natural representation of the plant which can be used for instance in models that are intended to describe the interaction of the plant and its environment (e.g. light) at a detailed level

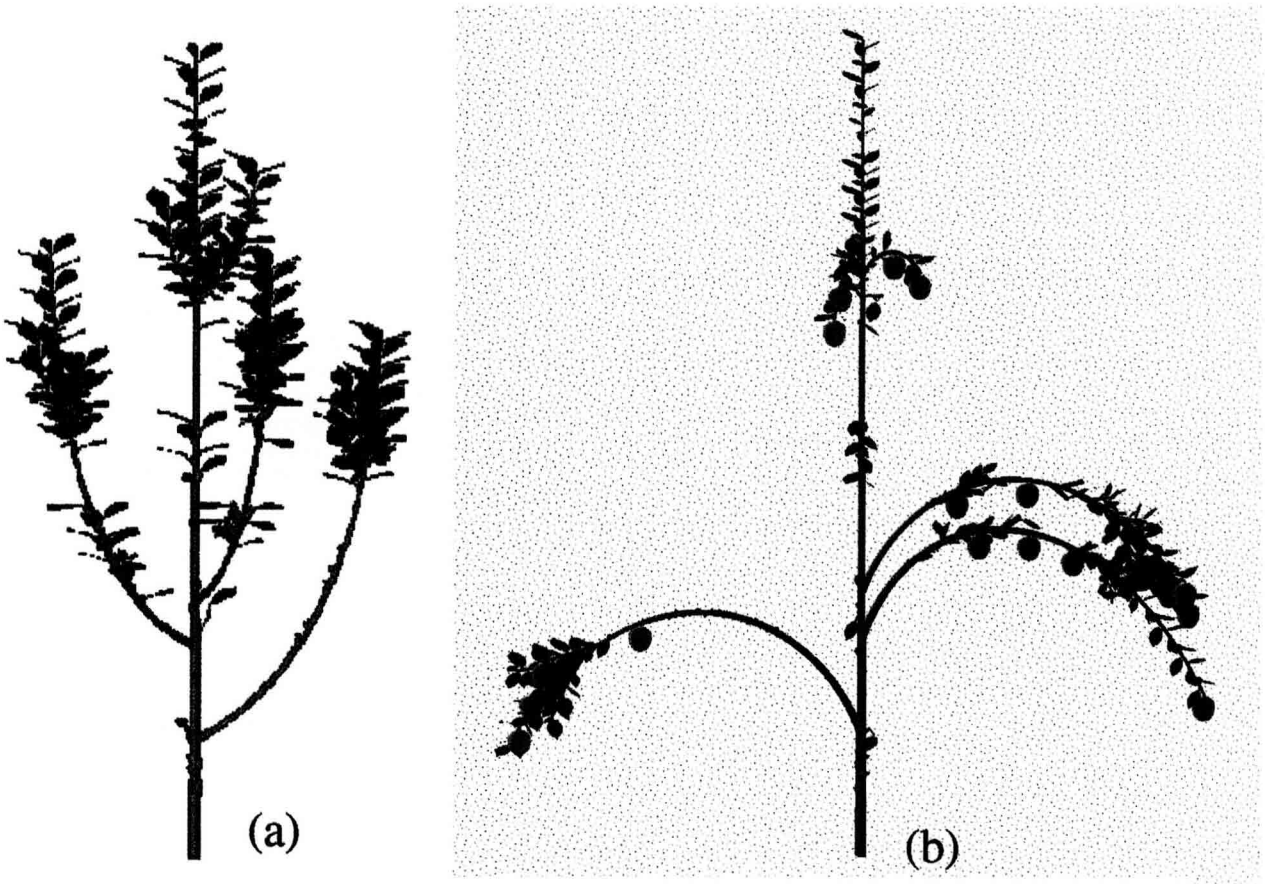


Figure 4. Virtual 3D reconstruction of the geometry of a plant with positioning of leaves (a) and fruits (b).

[32]. More generally, the user can plot a set of plants from the database (*figure 5*):

```
AML>          orchard          =
PlantFrame(plant_list)
AML> Plot(orchard)
```

3.3. Extraction of data samples

Visualising informations projected onto the 3D representation of plants is one way to explore the database. More quantitative explorations can be carried out and the most simple of these consists in studying how specific characters are distributed in the architecture of the plant population. To do this, samples of components are created corresponding to some topological or morphological criteria, and the distributions of one or several characters (target characters) are studied on this sample. This data extraction always follows the three following steps.

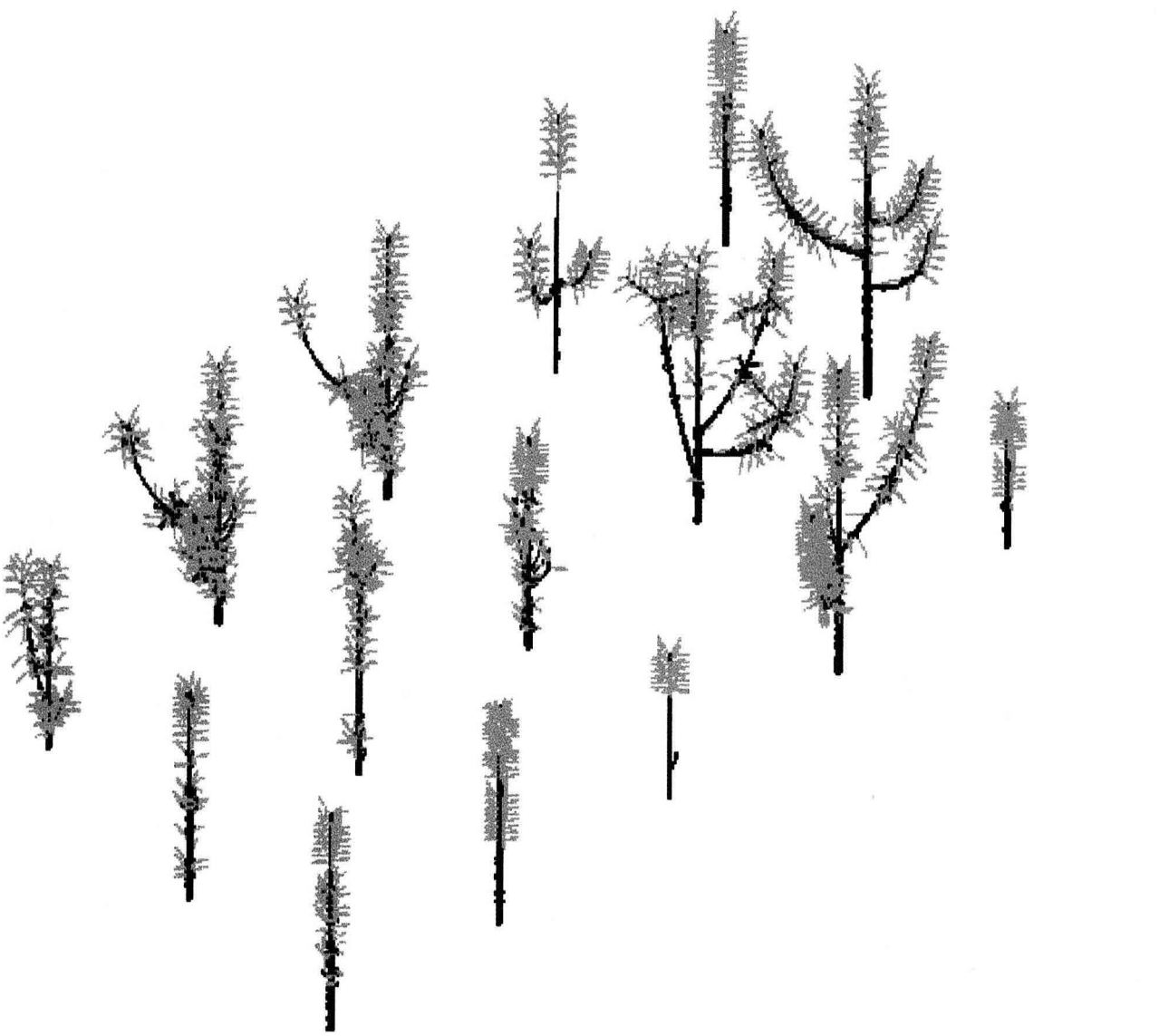


Figure 5. 3D representation of the information contained in the architectural database (only 15 plants are represented out of 102).

- First, a sample of components is created to study the target character.
- Second, the character itself is defined. It may be more or less directly derived from the data recorded in the field. For example, it is straightforward to define the diameter of a component if this has been measured in the field. On the other hand, the maximum branching order of the components that are borne by a given component needs some computation.
- Third, the target character is computed for each component of the selected sample of components.

The output of these three operations is a set of values that can be analysed and visualised in various ways. Let us assume for instance that we wish to determine the distribution of the number of internodes produced during a specific growth period for all the plants in the database. It is first necessary to determine the sample of components on which we wish to study this distribution. In our case, we assume that we are interested in the growth units of the trunk that are produced during the first year of growth. This would be written in AML as:

```
AML> sample = Foreach _component In
     growth_unit_list :
     Select(_component,
            Order(_component) == 0
            And Index(_component) == 90)
```

The variable sample thus contains the set of growth units whose order is 0 (i.e. which are parts of trunks) and whose growth year is 1990 (assuming 1990 corresponds to the first year of growth). The second step consists in defining the target character. This can be achieved by defining a corresponding function:

```
AML> nb_of_internodes(_x) = Size
(Components(_x))
```

The number of internodes of a component _x (assumed to be a growth unit) is defined as the size of the set of components that compose this growth unit _x (assuming that growth units are composed of internodes). Finally, this function is applied to each component in the previously selected sample and the corresponding histogram is plotted (*figure 6*):

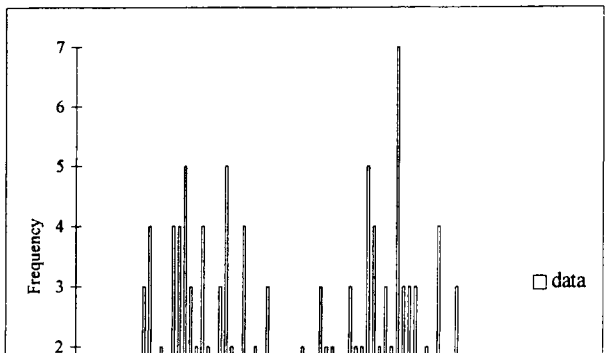


Figure 6. Histogram of the number of growth unit internodes for year 90 on the trunk.

```
AML> sample_values = Histogram
     (Foreach _component In sample
     :nb_of_internodes(_compo-
     nent))
AML> Plot(sample_values)
```

This example illustrates the kind of interaction a user may expect from the exploration of tree architecture. In the field, the growth units of the trunks produced during the first year of growth present a variable length, ranging roughly from 10 to 100 internodes. However, the quantitative exploration of the database shows that the histogram exhibits two relatively well-separated sub-populations of growth units (*figure 6*). The sub-population of short growth units corresponds to the first annual shoots of the trunk, made up of two successive intra-annual growth units, while the sub-population of long growth units corresponds to the first annual shoots made up of a single growth unit.

In order to separate and characterise these two sub-populations, we can make the assumption that

the global distribution is a mixture of two parametric distributions, more precisely, two negative binomial distributions. The parameters of this model can be estimated from the above histogram as follows:

```
AML>mixture = Estimate(sample_
    value, "MIXTURE", "NEGATIVE_
    BINOMIAL", "NEGATIVE_BINOMI-
    AL")

AML>Plot(mixture)
```

For all parametric models in the system, the function Estimate performs both parameter estimation and computation of various quantities (likelihood of the observed data for the estimated model, theoretical characteristics, etc.) involved in the validation stage. As demonstrated by the cumulative distribution functions in *figure 7b*, the data are well fitted by the estimated mixture of two negative binomial distributions. The weights of the two components of the mixture are very close (0.49/0.51), the first being centred on 21 internodes and the second on 53 internodes (*figure 7a*). Due to the small overlap of these two mixture components (*figure 7a*), the extracted sample can be optimally split up into two optimal sub-populations with a threshold fixed at 37.

As illustrated in this example, using AMAPmod, the user can query the database, make assumptions and look for data regularities. This interactive exploration process enables the user to build a rich and detailed mental representation of the architectural database, which relies on various complementary viewpoints.

3.4. Extraction and analysis of biological sequences

The previous section illustrates the extraction of a simple sample type, made up of numeric values. In this section, we consider a more complex sample type, made up of sequences of values. For example, in the apple tree database, let us consider sequences of lateral productions along trunks. Our aim is to analyse how lateral branches are distributed along the trunks of hybrids.

The sequences are coded as follows: for each plant, the 90 annual shoot of the trunk is described node by node from the base to the top. Each node

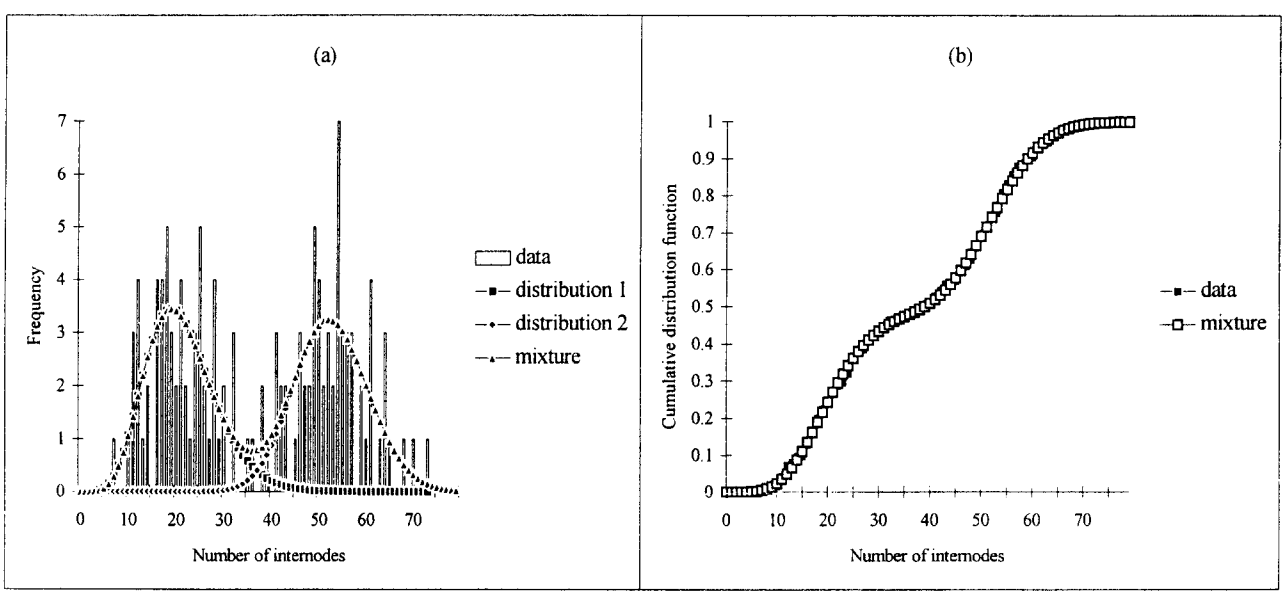


Figure 7. Modelling the number of growth unit internodes for year 90 on the trunk by a mixture of two parametric distributions.

is qualified by the type of lateral production (latent bud: 0, 1-year-delayed short shoot: 1, 1-year-delayed long shoot: 2 and immediate shoot: 3). This sample of sequences is built as follows in AML:

```
AML> seq =Foreach _component In
          annual_shoots_sample :
          Foreach _node In
          Axis(_component,
          Scale -> 4) :
          Switch lateral_type(_node)
          Case BUD: 0
          Case SHORT: 1
          Case LONG: 2
          Case IMMEDIATE: 3
          Default: Undef
```

The AML variable `annual_shoot_sample` contains the set of growth units of interest (assumed to be selected beforehand). For each component in this set, the array of nodes that compose its main axis is browsed by the second `Foreach` construct. Finally, for each node, a function `lateral_type()` (defined elsewhere) is used to encode the nature of the lateral production at that node.

Figure 8 illustrates the diversity of annual shoot branching structures encountered in the studied hybrid family, which results from the different branching habits of the two parents. In our context, we wish to characterise and classify the hybrids according to their branching habits. The difficulty arises from the fact that the branching pattern is made of a succession of branching zones which are not characterised by a single type of lateral production but by a combination of types (e.g. short shoots interspersed with latent buds). We shall use this example to illustrate how parametric models may be used in AMAPmod to identify and characterise successive branching zones along these annual shoots.

We assume that sequences have a two-level structure, where annual shoots are made up of a succession of zones, each zone being characterised by a particular combination of lateral production types. To model this two-level structure, we use a hierarchical model with two levels of representation. At the first level, a semi-Markov chain (Markov chain with null self-transitions and explicit state occupancy distributions) represents the succession of zones along the annual shoots and the lengths of each zone [4, 21, 22]. Each zone is represented by a state of the Markov chain and the succession of zones are represented by transitions between states. The second level consists of attaching to each state of the semi-Markov chain a discrete distribution which represents the lateral productions types observed in the corresponding zone. The whole model is called a hidden semi-Markov chain [19, 20].

The model parameters are estimated from the extracted sample of sequences by the function Estimate:

```
AML> hsmc = Estimate(seq,
"HIDDEN_SEMI-MARKOV",initial_hsmc,
segmentation -> True)
```

The first argument seq represents the extracted sequences, `"HIDDEN_SEMI-MARKOV"` specifies the family of models and `initial_hsmc` is an initial hidden semi-Markov chain which summarises the hypotheses made in the specification stage. An optimal segmentation of the sequences is required by the optional argument `Segmentation` set at `True` (this is a by product of the estimation of a model parameters).

The hidden semi-Markov chain built from the 90 annual shoots of the 102 hybrids is depicted in *figure 9* with the following convention: each state is represented by a box numbered in the lower right corner. The possible transitions between states are represented by directed edges with the attached probabilities noted nearby. Transient states are surrounded by a single line while recurrent states are surrounded by a double line. State i is said to be recurrent if starting from state i, the first return to state i always occurs after a finite number of transitions. A non-recurrent state is said to be transient. The state occupancy distributions which represent the length of the zones in terms of number of nodes are shown above the corresponding boxes. The possible lateral productions observed in each zone are indicated inside the boxes, the font sizes being roughly proportional to the observation probabili-

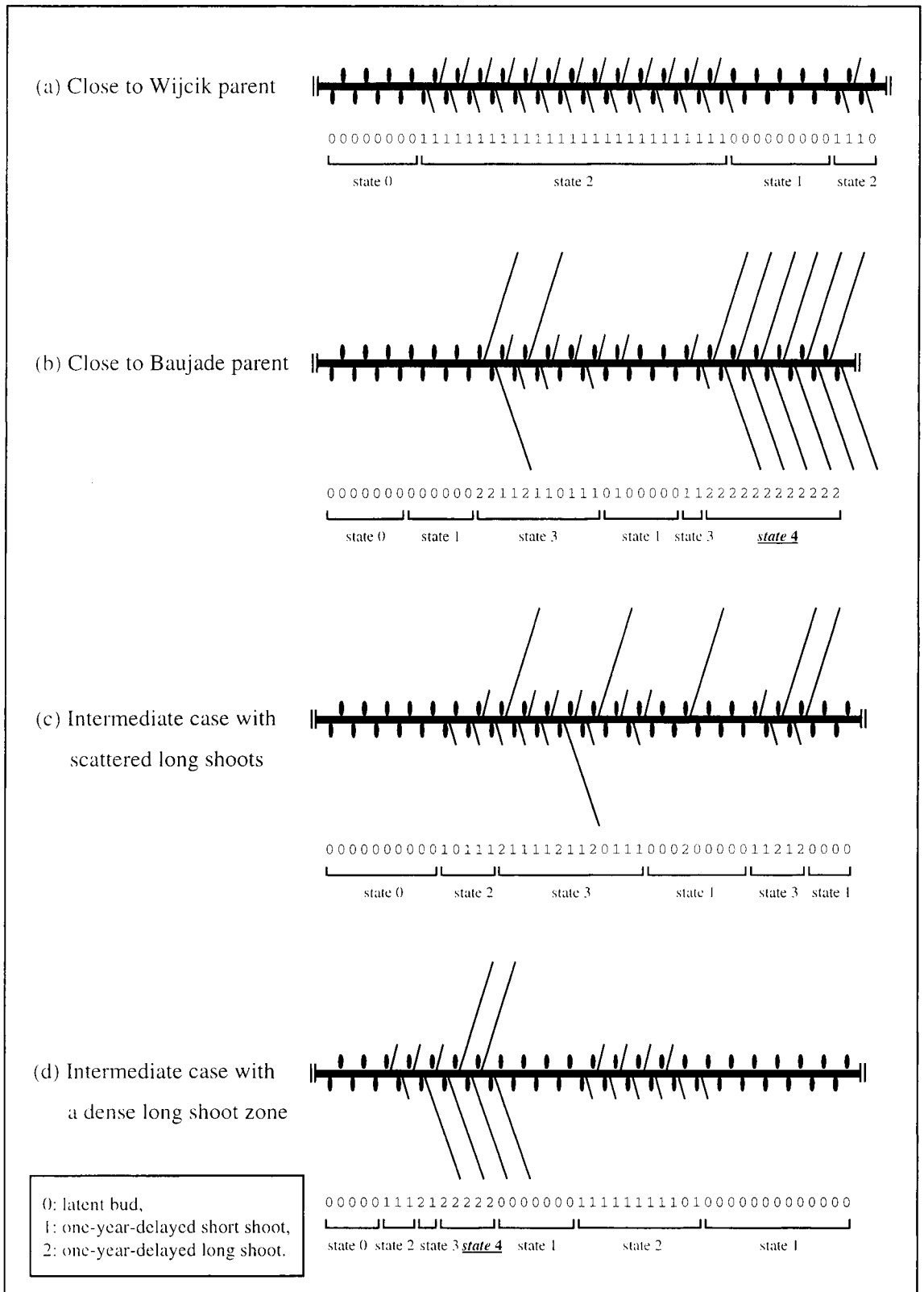


Figure 8. Example of sequences showing different branching habits in the hybrid family.

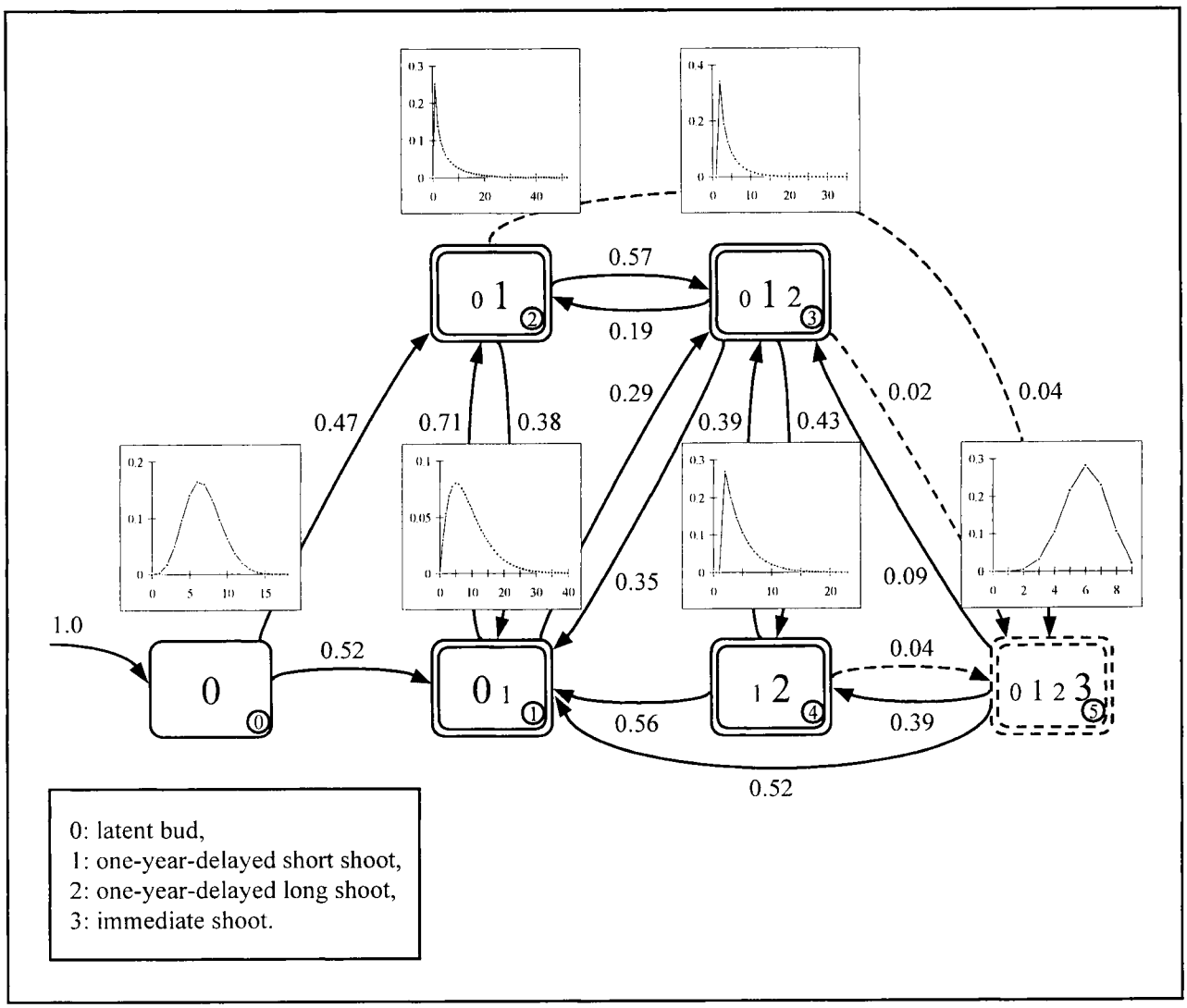


Figure 9. Hidden semi-Markov chain built from the 90 annual shoots of the 102 hybrids. Only transitions whose probability ≥ 0.02 are represented. The less probable transitions (respectively states) are represented by dotted edges (respectively, dotted boxes).

ties (for state 3, these probabilities are 0.1, 0.62 and 0.28 while for state 4, these probabilities are 0.01, 0.07 and 0.92 for latent bud, 1-year-delayed short shoot and 1-year-delayed long shoot, respectively). State 0 which is the only transient state is also the only initial state as indicated by the edge pointing towards state 0. State 0 represents the basal non-branched zone of the annual shoots. The remaining five states constitute a recurrent class which corresponds to the stationary phase of the sequences.

Building a parametric model gives us a global insight into the structure of the 90 annual shoot of the trunk for the 102 hybrids. The adequacy of the estimated model to the data is checked by examining the fitting of theoretical characteristic distributions computed from the model parameters to the corresponding observed characteristic distributions extracted from the data. Counting characteristic distributions for example focus on the number of occurrences of a given feature per sequence. The two features of interest are the number of series (or

clumps) and the number of occurrences of a given lateral production type per sequence. The fits of counting distributions (*figure 10*) can be plotted by the following function:

```
AML> Plot(hsmc, "Counting")
```

In addition, the optimal segmentation of the observed sequences in successive zones (*figure* 8) can be extracted from the model as a by product of the estimation of model parameters by the following function:

```
AML> segmented_seq = ExtractData(hsmc)
```

`segmented_seq` represents the observed sequences augmented by a variable which contains the corresponding optimal state sequences (*figure 8*). A careful examination of this optimal segmentation help us to highlight a discriminating

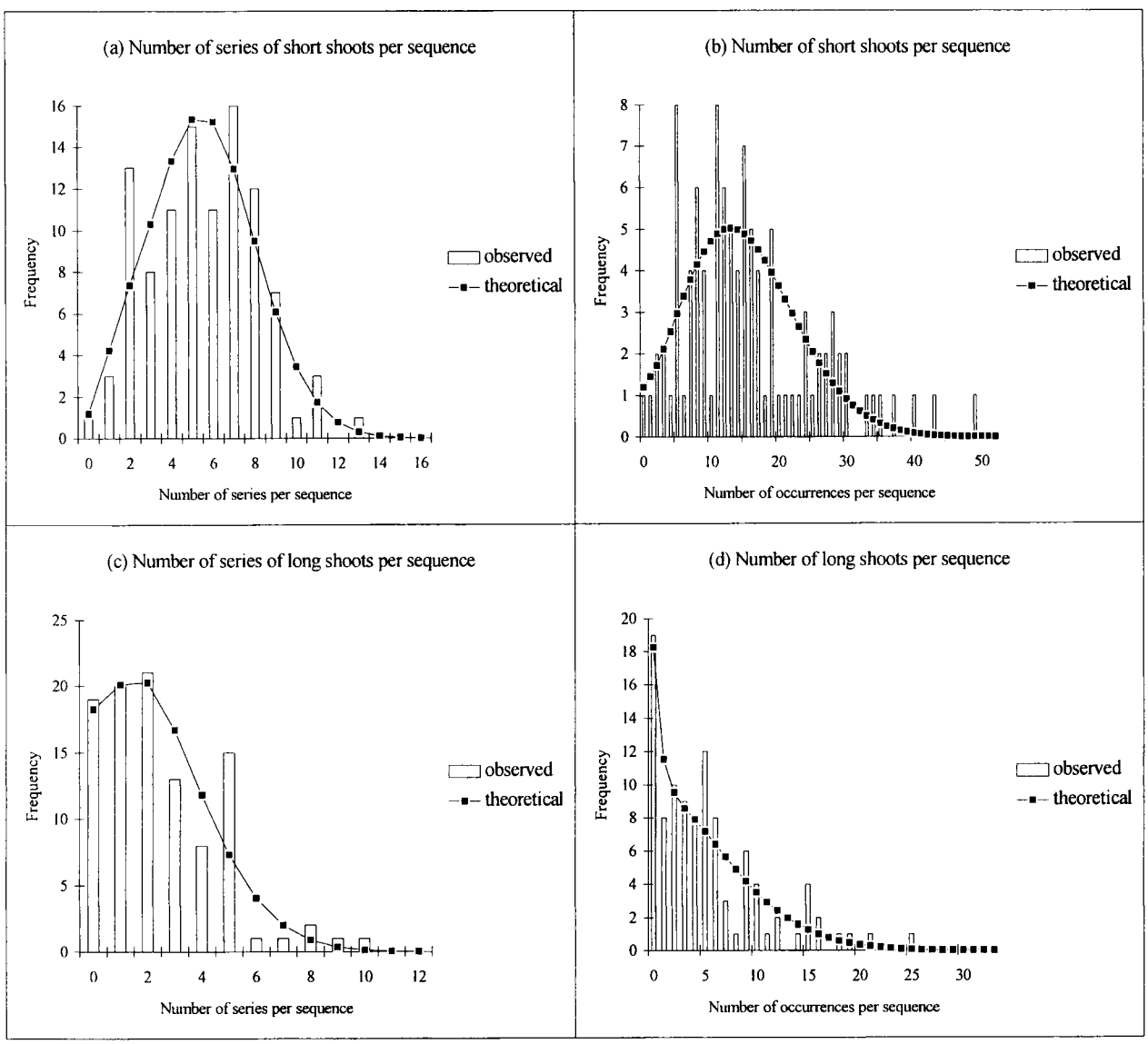


Figure 10. Fit of theoretical characteristic distributions computed from the model parameters to the corresponding observed distributions extracted from the data.

property: it suggests using the absence of state 4 in this optimal segmentation as a discrimination rule between hybrids closer to the Wijcik parent than to the Baujade parent (and conversely). State 4 corresponds to a dense long branching zone characteristic of the Baujade parent. Two sub-populations close to each of the parents are extracted by the function ValueSelect relying on the absence/presence of state 4 on the first variable:

```
AML> wijcik_seq = ValueSelect(segmented_seq, 1, 4, Mode -> Reject)

AML> baujade_seq = ValueSelect(segmented_seq, 1, 4, Mode -> Keep)
```

Simply counting the number of axillary long shoots per sequence would not have been sufficient, since for a given number of long shoots, these can be either scattered (*figure 8c*) or aggregated in a dense zone (*figure 8d*). This is confirmed by comparing the empirical distributions of the number of series with the number of occurrences of axillary long shoots per sequence extracted from the two hybrid sub-populations. The empirical distributions of the number of series / number of occurrences of axillary long shoots (coded by 2) per sequence for the sub-population close to the Wijcik parent can be simultaneously plotted by the following function (*figure 11a*):

```
AML> Plot(ExtractHistogram
        (wijcik_seq,
        "NbSeries", 2, 2),

     ExtractHistogram
        (wijcik_seq,
        "NbOccurrences", 2, 2))
```

These empirical distributions are very similar for the sub-population close to the Wijcik parent (*figure 11a*). Most of the series are thus composed of a single long shoot. These empirical distributions are very different for the sub-population close to the Baujade parent (*figure 11b*). In this case, the series are frequently composed of several successive long shoots.

The studied sample of sequences encompasses a broad spectrum of branching habits ranging from the Wijcik to the Baujade parent one. Hence, the building of a parametric model is mainly used for identifying a discrimination rule to separate the initial sample of branching sequences into two subsamples.

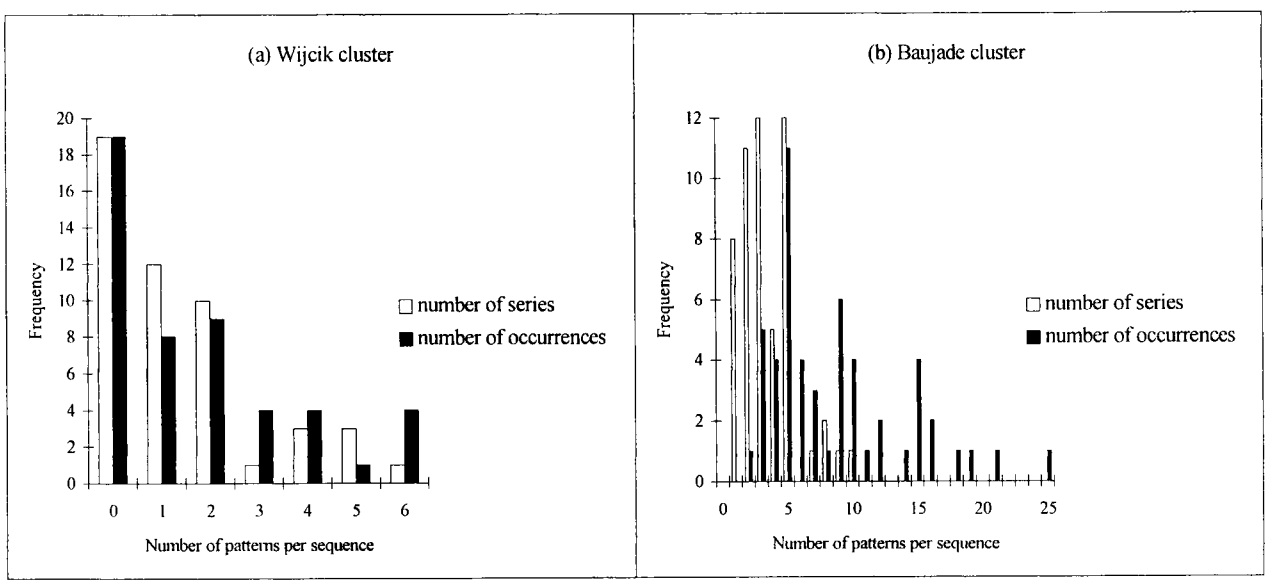


Figure 11. Number of series/occurrences of long shoots per sequence for the two sub-populations close to each parent.

4. Conclusions

In order to better control economical outcomes, a new trend in agronomic research consists in determining how production variables (e.g. wood biomass and quality, fruit quantity and quality) are distributed within plant architecture. This paper described the AMAPmod system which defines and implements a methodology dedicated to the investigation of plant architecture. This software has been available to the scientific community for 4 years now and has been used in the analysis of various types of plant architecture databases [1, 4, 13, 21, 22, 37]. It consists of about 200 000 lines of C++ code and its modular architecture makes it possible to add specialised modules.

The statistical inference approach developed in AMAPmod uses models that can account for the structural information contained in the MTG. The analysis of the information contained in a MTG requires the use of different techniques applied at different scales, dates and parts of plants. In view of the structural nature of the MTG, most of these techniques incorporate structural components (e.g. the graph of possible transitions for the hidden semi-Markov chain in (*figure 9*)).

The need for a common language to describe the architecture of a wide range of plant species and the will to share experience and tools in the exploration of plant architecture databases have been major motivations in the development of AMAPmod. Our long-term objective is to provide the agronomic community with a database management system that could be used as a standard tool. This standardisation process concerns various stages of AMAPmod methodology: plant architecture representation, its coding language, field observation protocols for given agronomic applications, macro-functions for database exploration (3D visualisation, sample extraction, etc.), tools for analysing structured data, etc. Such standardisation would facilitate the constitution and diffusion of plant corpora and would enable modellers to compare their models on the basis of publicly available databases.

Acknowledgments: We thank J.M. Lespinasse, S. Laurens, F. Delort and L. Fouilhoux for their contribution to the data collection.

References

[1] 11e colloque sur les recherches fruitières, Architecture et modélisation en arboriculture fruitière, Inra-Ctilf, Montpellier, France, 1998.

[2] Bouchon J., de Reffye P., Barthélémy D. (Eds.), Modélisation et simulation de l'architecture des végétaux, Inra éditions, Paris, 1997.

[3] Cilas C., Guédon Y., Montagnon C., de Reffye P., Analyse du suivi de croissance d'un cultivar de caféier (*Coffea canephora* Pierre) en Côte d'Ivoire, in: 11e colloque sur les recherches fruitières, Architecture et modélisation en arboriculture fruitière, Inra-Ctifl, Montpellier, France, 1998, pp. 46–55.

[4] Costes E., Guédon Y., Modelling the sylleptic branching on one-year-old trunks of apple cultivars, J. Am. Soc. Hortic. Sci. 122 (1997) 53–62.

[5] Costes E., Sinoquet H., Godin C., Kelner J.J., 3D digitizing based on tree topology : application to study the variability of apple quality within the canopy, Acta Hort. (1999) in press.

[6] de Reffye P., Modèle mathématique aléatoire et simulation de la croissance et de l'architecture du caféier Robusta : 3e partie. Étude de la ramification sylleptique des rameaux primaires et de la ramification proleptique des rameaux secondaires, Café Cacao Thé 26 (1982) 77–96.

[7] de Reffye P., Edelin C., Françon J., Jaeger M., Puech C., Plant models faithful to botanical structure and development, in: Dill J. (Ed.), Proceedings of SIGGRAPH'88, ACM, Atlanta, 1988, pp. 151–158.

[8] de Reffye P., Houllier F., Blaise F., Barthélémy D., Dauzat J., Auclair D., A model simulating above- and below-ground tree architecture with agroforestry applications, Agrofor. Syst. 30 (1995) 175–197.

[9] Dempster A.P., Laird N.M., Rubin D.B., Maximum likelihood from incomplete data via the EM algorithm (with discussion), J. R. Stat. Soc., Ser. B 39 (1977) 1–38.

[10] Ferraro P., Godin C., Un algorithme de comparaison d'arborescences non ordonnées appliqué à la comparaison de la structure topologique des plantes, in: Actes de SFC' 98, Société française de classification, 1998, pp. 77–81.

[11] Fisher J.B., Weeks C.L., Tree architecture of Neea Nyctaginaceae: geometry and simulation of branches and the presence of two different models, Bull. Mus. Hist. Nat. 7 (1985) 385–401.

[12] Fitter A.H., An architectural approach to the comparative ecology of plant root systems, New Phytol. 106 (suppl.) (1987) 61–77.

[13] Fournier D., Guédon Y., Costes E., A comparison of different fruiting shoots of peach trees, in: IVth International Peach Symposium, Bordeaux, France, 1998, pp. 557–565.

[14] Frijters D., Lindenmayer A., Developmental descriptions of branching patterns with paraclidial relationships, in: Rozenberg G., Lindenmayer A. (Eds.), Formal Languages, Automata and Development, North-Holland Publishing Company, Noordwijkerhout, The Netherlands, 1976, pp. 57–73.

[15] Godin C., Bellouti S., Costes E., Restitution virtuelle de plantes réelles : un nouvel outil pour l'aide à l'analyse de données botaniques et agronomiques, in: Proceedings of the Interface to Real and Virtual Worlds, Montpellier, France, 1996, pp. 369–378.

[16] Godin C., Caraglio Y., A multiscale model for plant topological structures, J. Theor. Biol. 191 (1998) 1–46.

[17] Godin C., Costes E., Caraglio Y., Exploring plant topological structures with the AMAPmod software: an outline, Silva Fennica 31 (1997) 357–368.

[18] Godin C., Guédon Y., Costes E., Caraglio Y., Measuring and analyzing plants with the AMAPmod software, in: Michalewicz M.T. (Ed.), Plants to Ecosystems. Advances in Computational Life Sciences, I, CSIRO Publishing, Melbourne, 1997, pp. 53–84.

[19] Guédon Y., Analyzing nonstationary discrete sequences using hidden semi-Markov chains, Document de travail, Montpellier, France, Cirad, 1998, Report no. 5–98.

[20] Guédon Y., Hidden semi-Markov chains: a new tool for analyzing nonstationary discrete sequences, in: 2nd International Symposium on Semi-Markov Models: Theory and Applications, Compiègne, France, 1998.

[21] Guédon Y., Barthélémy D., Caraglio Y., Analyzing spatial structures in forest tree architectures, in: Proceedings of the IUFRO Workshop Empirical and Process-Based Models for Forest Tree and Stand Growth Simulation, Oeiras (Portugal), 1999, in press.

[22] Guédon Y., Costes E., A statistical approach for analyzing sequences in fruit tree architecture, Acta Hort. (1999) in press.

[23] Hallé F., Oldeman R.A.A., Essai sur l'architecture et la dynamique de croissance des arbres tropicaux, Masson, Paris, France, 1970.

[24] Hallé F., Oldeman R.A.A., Tomlinson P.B., Tropical Trees and Forests. An Architectural Analysis, Springer-Verlag, New York, 1978.

[25] Hanan J.S., Room P., Practical aspects of plant research, in: Michalewicz M.T. (Ed.), Plants to Ecosystems. Advances in Computational Life Sciences, I, CSIRO publishing, Melbourne, 1997, pp. 28–43.

[26] Harper J.L., Rosen B.R., White J., The Growth and Form of Modular Organisms, The Royal Society, London, 1986.

[27] Honda H., Description of the form of trees by the parameters of the tree-like body : Effects of the branching angle and the branch length on the shape of the tree-like body, J. Theor. Biol. 31 (1971) 331–338.

[28] Honda H., Tomlinson P.B., Fisher J.B., Two geometrical models of branching of tropical trees, Ann. Bot. 49 (1982) 1–12.

[29] Jackson J.E., Palmer J.W., Light distribution in discontinuous canopies: calculation of leaf areas and canopy volumes above defined irradiance contours for use in productivity modelling, Ann. Bot. 47 (1981) 561–565.

[30] Mitchell K.J., Dynamics and simulated yield of Douglas fir, For. Sci. 21 (1975) 1–39.

[31] Prusinkiewicz P., Remphrey W.R., Davidson C.G., Hammel M.S., Modeling the architecture of expanding *Fraxinus pennsylvanica* shoots using L-systems, Can. J. Bot. 72 (1994) 701–714.

[32] Rapidel B., Étude expérimentale et simulation des transferts hydriques dans les plantes individuelles. Application au Caféier *Coffea arabica* L, Ph.D. thesis, USTL Montpellier France, 1995, 246 p.

[33] Remphrey W.R., Neal B.R., Steeves T.A., The morphology and growth of Arctostaphylos uva-ursi bearberry: an architectural model simulating colonizing growth, Can. J. Bot. 61 (1983) 2451–2458.

[34] Ross J.K., The radiation regim and the architecture of plant stands, Junk W. Pubs., The Hague, The Netherlands, 1981.

[35] Sabatier S., Ducouso I., Guédon Y., Barthélémy D., Germain E., Structure de scions d'un an de Noyer commun, *Juglans regia*, variété Lara greffés sur trois porte-greffe (*Juglans nigra, J. regia, J. nigra* × *J. regia*), in: 11e colloque sur les recherches fruitières, Architecture et modélisation en arboriculture fruitière, Montpellier, France, 1998, pp. 75–84.

[36] Sinoquet H., Adam B., Rivet P., Godin C., Interactions between light and plant architecture in an agroforestry walnut tree, Agrofor. Forum 8 (1997) 37–40.

[37] Sinoquet H., Godin C., Costes E., Mesure de l'architecture des plantes par digitalisation 3D, in: Numerisation 3D/Human modelling 98, Uic, Paris, 1998.

[38] Sinoquet H., Rivet P., Godin C., Assessment of the three-dimensional architecture of walnut trees using digitizing, *Silva Fennica* 3 (1997) 265–273.

[39] Sinoquet H., Thanisawanyangkura S., Mabrouk H., Kasemsap P., Characterisation of the light environment in canopies using 3D digitising and image processing, Ann. Bot. 82 (1998) 203–212.