The Optimization of the 3-d Structure of Plants, Using Functional-Structural Plant Models. Case Study of Rice (*Oryza sativa* L.) in Indonesia

Dissertation
for the award of the degree
"Doctor rerum naturalium" (Dr.rer.nat.)
of the Georg-August-Universität Göttingen

within the doctoral program Environmental Informatics (PEI)
of the Georg-August University School of Science (GAUSS)

submitted by
Ditdit Nugeraha Utama

from Indonesia
Göttingen, 2015

Thesis Committee

Prof. Dr. Winfried Kurth
(Department Ecoinformatics, Biometrics and Forest Growth, University of Göttingen)
Dr. Katarina Streit
(Department Ecoinformatics, Biometrics and Forest Growth, University of Göttingen)
Prof. Heiko C. Becker
(Department of Crop Science, University of Göttingen)

Members of Examination Board

Prof. Dr. Winfried Kurth
(Department Ecoinformatics, Biometrics and Forest Growth, University of Göttingen)
Prof. Dr. Heiko C. Becker
(Department of Crop Science, University of Göttingen)

Further members of Examination Board

Prof. Dr. Kerstin Wiegand
(Department of Ecosystem Modelling, University of Göttingen)
Prof. Dr. Dieter Hogrefe
(Institute of Computer Science, Telematics Research Group, University of Göttingen)
Prof. Dr. Carsten Damm
(Theoretical Computer Science and Algorithmic Methods, University of Göttingen)
Jr.-Prof. Dr. Anja Fischer
(Institute for Numerical and Applied Mathematics, University of Göttingen)

Date of the oral examination: 30.11.2015

Acknowledgements

I would like to thank:

- Prof. Dr. Winfried Kurth - for his patience, many motivating discussions, enthusiasm, and immense knowledge. His deep insights helped me at various stages of my research.
- Prof. Dr. Heiko C. Becker - for his insightful comments and encouragement.
- Dr. Katarina Streit - for her comments and suggestions.
- Mamih and Papih - for love, prayer, and infinite support throughout everything.
- Mimi and Mama - for love, support, and prayer.
- Ade - for love, prayer, and supporting me spiritually throughout writing this thesis and my life in general.
- Teh Rinrin, Teh Hanny, Teh Vinni - for love, support and prayer for me.
- Kak Topik, Kang Acep - for love, support and prayer for me.
- Kak Uun, A Asep, Adi, Bambang - for love, support and prayer for me.
- All my nieces and my nephews - for love, support and prayer for me.
- Yong and Michael - for being friends and partners in discussion.
- Ms. Ilona Watteler-Sprang and Dr. Reinhold Meyer - for the excellent administrative and technical support in our department.
- Everyone who I met during my PhD program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

There are three motivations of the research efforts presented in this thesis. The first is regarding the performance of plants. The performance of plants is directly and indirectly affected by the 3-d physical structure of plants [175]. One capability considered as an indicator of plants' performance is the capability to intercept the photosynthetically active radiation (PAR). A lot of combinations of plant 3-d physical structure parameters that correspond to the capability of PAR interception have to be analyzed. This approach probably can be used to morphologically configure an ideal type (ideotype) of a plant [104]. Conducting research about the capability of plants to intercept the light is the first motivation.

Furthermore, functional-structural plant modelling (FSPM) has become one popular approach used in several researches about plants and forests in recent years, where the protection of the environment and aspects of climate change have been hot issues for numerous research topics. FSPM technically addresses three kinds of structural aspects which strongly relate with plant architecture, internal, functional and environmental impacts [134] that can be simulated in virtual environments accessible by 3-d views [147] [172] [78]. Conducting research about optimization for 3-d structural architecture of plants by using FSPM is one motivation itself. The parameters of the 3-d physical structure of a plant that interconnect to each other and possibly can be optimized to improve the performance of plants are analyzed deeply.

The last motivation is concerning the rice plant. It is well known that the rice plant is the most popular and important crop plant in several countries in the world [58]. The rice is Indonesia's staple food as well [174]. As an agricultural activity, the rice production is

significantly affected by the local climate condition [108]. Thus, the research about the correlation between the morphological rice plant performance and Indonesian climate conditions reasonably becomes the third motivation of the research.

1.1.1 Research Questions

Research questions are critical issues for guiding research. There are three types of research questions which are addressed in this research. They are:

1. What are the reasons why rice plants have their specific physical structure?
2. Among parameters, which is the most influential on the architecture and shape of rice plants, and how sensitive is the functioning of the plant to changes of the parameter values?
3. How can we design improved virtual rice plants by using FSPM?

1.1.2 Research Objectives

According to the research questions, the general goal of the research is defined. It is to develop an optimization model of the 3-d physical structure for rice plants by using FSPM. Three types of specific research objectives are:

1. To analyze and describe the reasons why rice plants have their specific physical structure.
2. To analyze and describe the requirements that rice plants must meet to optimize their forms.
3. To design and construct a 3-d model of virtual rice plants by using FSPM on the software platform GroIMP with the possibility to optimize form with respect to performance.

1.2 Thesis Organization

To deliver the result of the research, except the introduction chapter, the thesis consists of four other parts. They are in that order presented: fundamental theoretical framework, research methodology, constructed model configuration and result, and discussion, conclusion and further works.

The part "fundamental theoretical framework" consists of the chapters plant modelling and the language XL, fundamental facts about the rice plant, and algorithms for optimization. The chapter "plant modelling and the language XL" presents four subsections; the functional-structural plant models, an approach for modelling the plants; rewriting systems, a technique to define a complex object; turtle concept, as a concept to interpret a rewriting technique in geographical purpose; and basic XL programming with GroIMP, as a computer language that is executed under a model platform software. The chapter "fundamental facts about the rice plant" talks about the life cycle and vegetative green organs of rice plants. And, the chapter "algorithms for optimization" mentions the algorithms of five optimization methods.

The part "research methodology" specifically mentions the methods used in the research. It talks about research methodology structure, empirical data, model development, and sensitivity analysis and simulation experiments. Especially in the part "constructed model and result", the detailed explanations of the constructed model (such as rice plant model, skylight model, parameters for optimization, and optimization model) and the results of model and simulation (such as rice plant morphology comparisons, sensitivity analysis, skylight model experiment, and optimization result) are delivered. Finally, the thesis is closed by the part "discussion, conclusion, and further works".

1.3 Modelling Plant Performance: State of the Art

1.3.1 Functional-Structural Plant Modelling

FSPM is a plant modelling approach which combines three types of model; morphological, physiological, and statistical models. The morphological models describe the plant's structure and development. The physiological models demonstrate the biological processes of the plant at a deeper level of causal and functional relationships. And the statistical models, also termed aggregation models, deal with the whole plant in a statistical approach [94]. Regarding an approach in modelling the plant, [12], referring to Aristotle, identified four types of causes. Those causes are material cause (substances), formal causes (form), efficient causes (external influences), and purposes. Especially for the fourth type of causes, it means that a plant grows the way it does by reference to the purpose or aim of the growth. For example, the plant is growing with wide leaves because it is trying to maximize its light interception [134]. A justification for this teleological (=purpose-related) viewpoint is evolution, which favours properties enhancing survival and reproduction.

The FSPM, or alternatively called virtual plant, constitutes a new generation of models that is able to represent many potentially important aspects of plant growth and function. FSPM also explicitly reflects the first three of Aristotle's causes. Structural aspects of plant architecture are concerned with formal cause, internal functional aspects are concerned with material cause, and environmental impacts are concerned with efficient cause.

The FSPM represents many important interacting processes in a dynamic way and at a high degree of detail. A dynamic realism is FSPM's strength and also its weakness, as the dynamic realism makes FSPM more complex and computationally demanding. It is also useful to investigate and provide insight into the last of Aristotle's causes of plant growth strategies (purposes-related causes of growth). A promising option to address the aspect of adaptive "purpose" is by using evolutionary optimization algorithms (EA) [47] [7]. The combination of EA and FSPM will be a good way to explore the plant structure and growth strategies in perspective on relationships among evolution, ecosystems, individual plants, and genes [129]. Basically, the idea is that FSPM parameters which define a growth strategy are assumed as genetic information that can change with evolution. The growth strategy parameters are used as a basis for developing a different set of parameters for each genotype in a population of genotypes. Then, phenotypic realization of each genotype is simulated by runs of the FSPM. The relative reproductive success of each phenotype is determined. Finally, the measures of relative reproductive success are used to generate a new population of genotypes [134].

Furthermore, two types of analysis method that are very important and needed in modelling plant growth and architecture are the qualitative botanical and quantitative statistical analysis. By using qualitative botanical analysis, the development sequence of a plant is studied by the identification of various levels of organization and of homogeneous sub-units. In contrast, all of these architectural units, e. g., axis and growth unit (results of particular growth processes) can be described by using the quantitative analysis approach. [38] have combined both approaches to create a simulation model for tree architecture with agroforestry applications. There are many quantitative data used in their research, such as: (1) growth processes: length of the growth units (GU), number of internodes in each GU, polycyclism, state of the apex at each GU (living, dead, broken), nature of the axis (dominant / dominated / forked); (2) branching processes: nature and position of lateral buds, axes or floral buds; (3) geometric aspects: branch diameter at the base of each GU, angle of insertion.

Some results are based on (a) statistical analysis and fitting of the different static frequency distributions of internodes number per different type of GU in many types of distribution, such Poisson, binomial and geometric distribution; (b) simulation of primary

growth, how the successive GUs are stacked can be known, the GUs can be characterized according to their physiological age; (c) simulation of radial growth, it simulates the apical meristems producing new leafy shoots and the diameter of vegetative axes increased by one new growth ring; (d) simulation of plant / plant and plant / environment interactions, it possibly simulates the growth of buds which are subject to global physical constraints (even if they belong to different plants). The volume in which the forest stand developed is discretized into elementary cubes ("voxels"), in which interactions can be modeled, leading to the simulation of competition for space between buds which grow within the same cube, whether they belong to the same plant or to neighboring plants; and (e) simulation of below-ground architecture, the model simulates that root development can occur in different ways, such as sylleptic or proleptic, partial or total, etc.

Concentrating on the evolution of plant shape, [119] simulated phenotypic walks through multi-dimensional fitness-landscapes by using a simple plant morphospace as a venue for simulated walks. In these computer simulated walks, Niklas used the limits of the morphological plasticity concept of [165] about the morphological variation in the fossil record of Upper Silurian and Devonian Period vascular land plants; such as: equal branching, tracheids, stomata, unequal branching, etc. Developmental plasticity itself can be defined as the developmental changes that follow the perception and integration of environmental information. Although developmental plasticity plays a major role in the adaptation of both animals and plants to heterogeneous conditions, it is thought to be of particular importance in plants [3] in [124].

[120] expanded and re-evaluated a model for mimicking land plant evolution; it is used to predict the effects of the number of simultaneously performed tasks (complexity), abrupt changes in environment conditions, and developmental barriers on number and accessibility of variants occupying fitness maxima. There are four types of task which must be performed: maximizing light interception (L), maximizing mechanical stability (M), maximizing reproductive success (R), and minimizing total surface area (S). Based on those types combinations, there are 15 fitness landscapes: four single-task landscapes (L, M, R, S), six double-task landscapes (MR, LM, RL, MS, LS, RS), four triple-task landscapes (LMR, MRS, MLS, LRS), and one four-task landscape (LMRS).

In the model, abrupt changes in environment conditions are mimicked by random replacement of one fitness landscape with another; on the other hand, the developmental barriers are mimicked by barring searches from entering specific subdomains in the morphospace. The morphospace is constructed with three parameters, they are: a bifurcation angle ϕ, the angle between the longitudinal axes of each bifurcate pair; a rotation angle γ, the angle between the longitudinal axis of each pair and the horizontal plane;

and a probability of apical bifurcation p. In addition, the simple Y-shaped variant is modified by the addition of more axes to simulate more complex morphological variants in the morphospace. The morphospace itself has two subdomains, one containing all equally branched (isobifurcate) morphologies and another containing all unequally branched (anisobifurcate) morphologies. The entire morphospace is constructed by independently varying each of the variables used to construct the morphologies in each of the two subdomains [122].

Based on the idea of five developmental processes generating all plant body plans [121]: the degree to which cyto and karyokinesis are synchronized; the extent to which dividing cells remain adjoined; whether cytoplasmic continuity is maintained after cell division; whether growth in size is determinate (closed) or indeterminate (open); and the number and orientation of the planes of cell division; [123] studied the evolutionary development of plant body plans. They deal with the 'transcription factor' paradigm helping to unravel the developmental macroevolution of plants. This paradigm has implicated at least six molecular mechanisms for phenotypic evolution: (1) gene array duplication and subsequent sub-functionalization, (2) changes in the spatial expression patterns of pre-existing arrays, (3) homeodomain protein sequence alterations, (4) modifications of DNA binding domains, (5) alterations in downstream regulated gene-networks, and (6) changes in upstream regulatory genes.

The real challenge in terms of modelling plant evolution is to assess the simultaneous performance of all of basic biological obligations. For that reason, [122] developed a computer model to mimic the early evolution of plants (in this case ancient vascular plants or tracheophytes are the objects of the research). The model has three components: an N-dimensional domain of all mathematically conceivable ancient morphologies (a morphospace); a numerical assessment of the ability (fitness) of each morphology to intercept light, maintain mechanical stability, conserve water, and produce and disperse spores; and an algorithm that searches the morphospace for successively more fit variants (an adaptive walk). The early land plant evolution is simulated by locating neighboring morphologies that progressively perform one or more tasks more efficiently.

The relative fitness of each hypothetical morphology can be evaluated using basic physics or engineering principles that describe quantitatively the performance of each task designated to influence growth, survival, and reproductive ability. The ability of each morphological variant to perform one or more of these tasks can then be divided by the maximum performance level in a particular landscape.

In a similar spirit, [130] made a model to describe and explain the evolution and development of inflorescence architectures; it could predict associations between inflorescence architecture, climate, and life history; which are validated empirically. Three types of

inflorescence architecture observed are: panicles, which comprise a branching series of axes that terminate simultaneously in flowers; racemes, which comprise axes bearing flowers in lateral positions, or lateral axes that reiterate this pattern; and cymes, which comprise axes that terminate in flowers and lateral axes that reiterate this pattern [158] [159] [177]. In modelling, [130] considered vegetativeness (veg), a variable that characterizes the meristems giving rise to shoots or flowers as two extremes of a continuum; where the veg level was categorized into two categories: high levels of veg corresponding to shoot meristem identity and low levels to flower meristem identity. These veg levels were related to many factors, for instance plant age, meristem position, internal state of a meristem, and the environment.

In generating the next architectural type, the meristem can be in one of two internal states (state A and B); state A represents an advanced stage of meristem development (mature), whereas state B presents the stage when a meristem is newly formed (immature). State B can be transient. In addition, meristems in state A and B attain low levels of veg at different times, T_A and T_B. Figure 1.1 depicts architectures and time courses of veg decline for various inflorescence models. Small filled circles indicate meristems; white circles denote flowers; and colors highlight paths of representative meristems: blue for main meristem, orange for lowest lateral meristem, and red for third lateral meristem from bottom. Plots show the time course of veg decline in selected meristems after their initiation. (A) Level of veg does not change with time; an indeterminate vegetative branching structure results. (B) Veg declines at a similar rate in all meristems and yields flowers upon reaching threshold V_K at time T; a panicle results. (C) Veg in apical meristems (state A) reaches threshold V_K at time $T_A > T_B$ for lateral meristems (state B). The resulting structure is a compound raceme, with lower branches terminating in flowers. (D) Transient model in which lateral meristems are initially in state B but revert to state A if veg does not reach the threshold V_K; a raceme with indeterminate branches is produced for $T_B < T_A$. (E) Transient model in which $T_B > T_A$ yields a cyme [130].

In the transient model, it is assumed that state B can be reverted to state A. The key feature of the transient model is that it can generate cymes as well as racemes and panicles, thus accounting for these inflorescence types within a single framework. The transient mechanism may therefore account for the restriction of observed inflorescence types to a small region of morphospace (Figure 1.2); where different phenotypes are generated by varying the two times, TA and TB at which flowers begin to form that correspond to panicles, racemes, and cymes; values along each axis range from 0 to 10 plastochrons; black arrows, pointing away from the wild-type architecture, indicate the effect of LFY and TFL1 mutations [141].

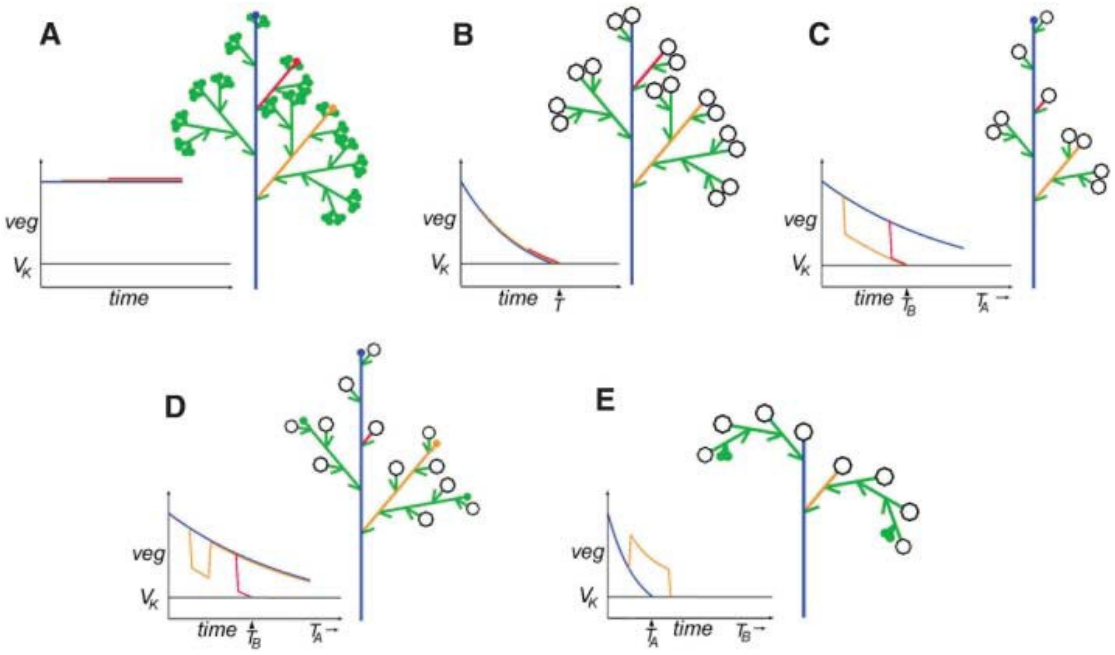

FIGURE 1.1: Architectures and time course of *veg* decline for various inflorescence models [130]

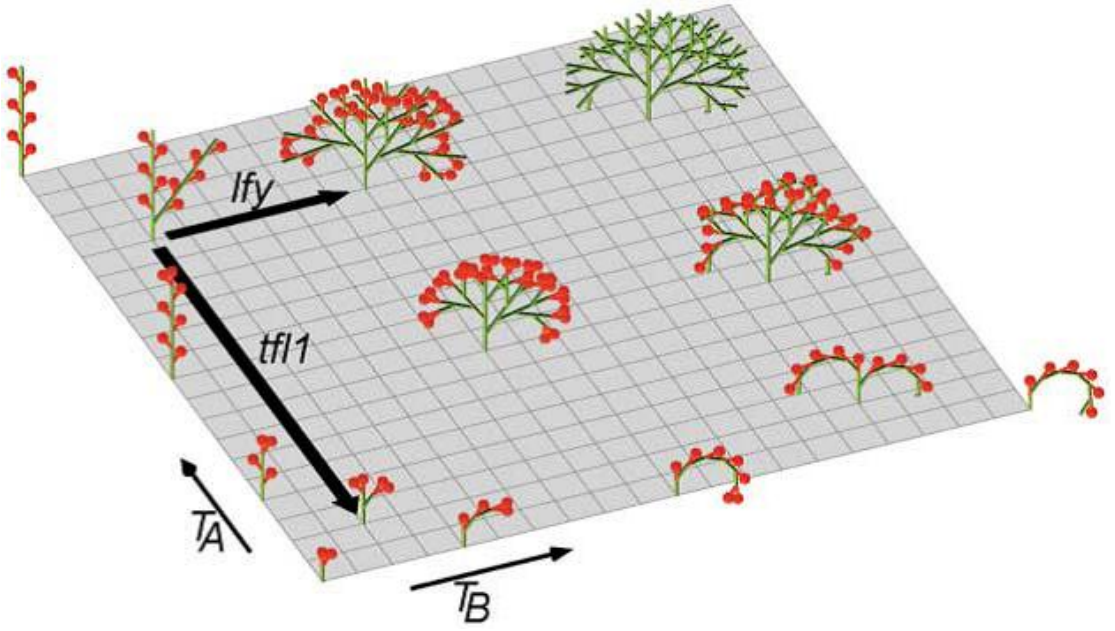

FIGURE 1.2: Morphospace for the transient model [130]

Figure 1.3 describes the incorporation of LFY and TFL1 genes into the transient model. (A) shows the interactions between genes, time, *veg*, and growth underlying the model. Growth increases the number of modules and hence influences the spatial pattern of gene expression. Gene activity affects *veg* and hence influences whether a meristem will continue to generate more modules or whether it will cease growing. Arrowheads indicate up-regulation; bars, down-regulation. Growth promotes production of meristems in state A or B, with state B reverting to A unless the floral threshold is reached. (B to H) show wild-type, mutant, and transgenic phenotypes generated by the model with the interactions shown in (A), assuming inductive conditions [141]. Moreover, circles indicate flowers; a color-code indicates *veg* levels: white for normal flower; yellow/green for shootlike flower; and arrows indicate branches.

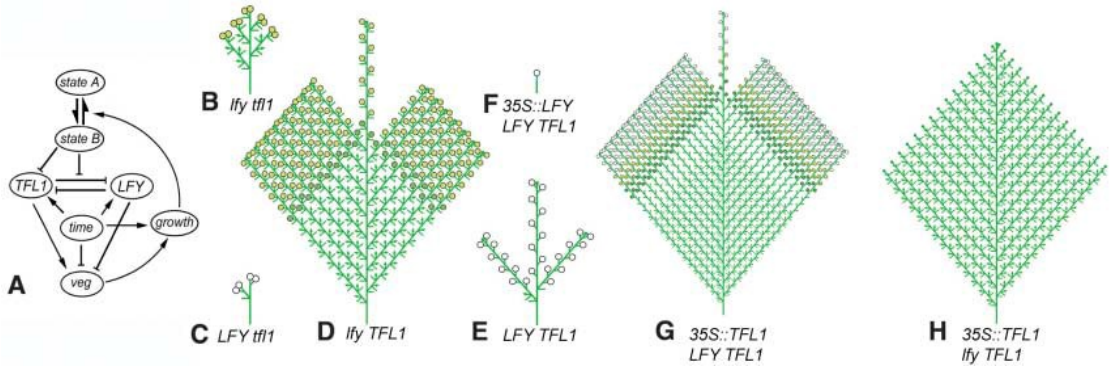

FIGURE 1.3: Incorporating LFY and TFL1 genes into the transient model [130] (see text for explanation)

Several two-dimensional fitness landscapes, with their forms depending on further parameters (T_d, σ, and θ), are depicted in Figure 1.4. Fitness levels are indicated by height and color. For each genotype, fitness is calculated over seasons with an average duration T_d and standard deviation σ, assuming that a fraction θ of mature plants survives from one season to the next. Plants illustrate the architecture generated at time T_d. (A) For annuals with fixed growth duration, the optimal inflorescence is a panicle, represented by a single adaptive peak. (B) If T_d is reduced, the optimal architecture is a less highly branched panicle. (C) When σ is increased; two peaks arise corresponding to compound racemes and cymes. (D) With a further increase in σ, the peaks diverge. Optimal architectures are simple racemes and cymes. (E) Increased longevity θ shifts the peaks toward panicles [130]. The separation of peaks in cases C, D, E can explain why racemes and cymes rarely co-occur within the same plant genus.

In another, more agriculture-oriented plant modelling study, [44] applied the crop model PILOTE and the FSPM GREENLAB for modelling maize (Zea mays L.) at the stand level and individual level. The crop model PILOTE was used to describe the plant as an object located within the surface area of a field responding to temperature, light,

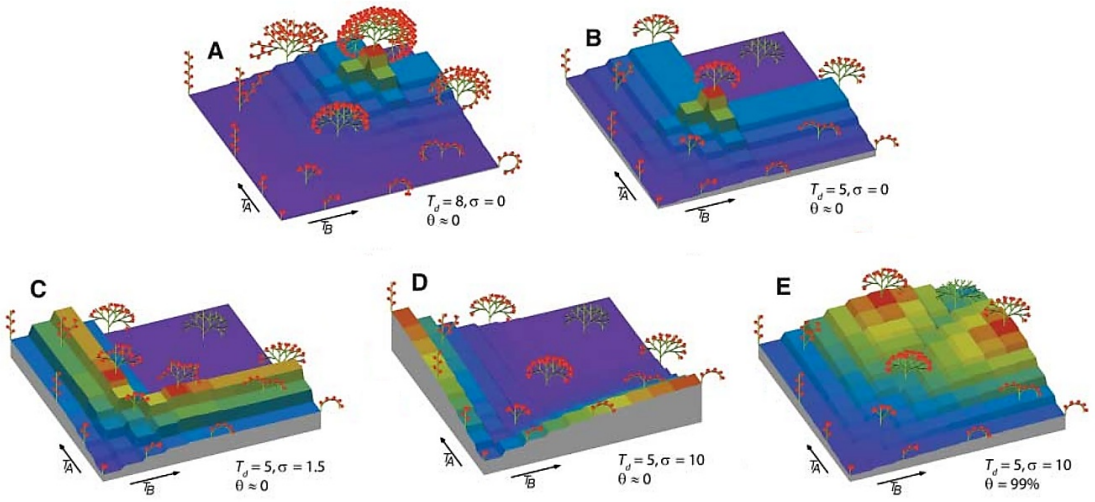

FIGURE 1.4: Two-dimensional fitness landscapes [130]

water dynamics and other factors influenced by humans [18] [107]. PILOTE has a soil model to simulate the soil water balance and a crop model to estimate the resulting crop yield. The simulation of the soil water balance, by determining evaporation, was used to see the effect of the current leaf area index (LAI) on the partitioning coefficient between transpiration and soil evaporation, calculating the soil water balance among reservoirs on basis of field capacity (FC) and wilting point (WP), and exporting water stress index (WSI) as an environmental coefficient to the other model (the crop module). On the other hand, the crop module simulates the LAI and WSI response by involving two shape parameters and a vegetative growth parameter correlating with the effective accumulated temperature. It evaluates grain yield as well as product and harvest index (HI).

Other issues virtual plants were studied by [116]. They conducted research in mechanics and form of leaves, especially in the leaf of maize plant (Zea mays cv DEA) flexural behavior. They used maize plants as objects of research which were raised from seeds during late spring and summer in Bordeaux. In conducting the research, they processed data based on a local structural definition of longitudinal leaf suppleness.

Theoretically, based on a mechanical perspective, the maize leaf can be described as a complex 3-d structure consisting of a central thickened, curved midrib; and a much thinner part, on both sides of the midrib, called lamina (Figure 1.5). Here, the study was focused on the bending behavior of the central line of the midrib; when the whole structure of the leaf is vertically loaded, this line remains within the vertical x-y plane (symmetry). Moreover, still based on the mechanical point of view, the bending suppleness can be defined in equation (1.1); where sf is the plane bending suppleness, $\delta(C)$ is

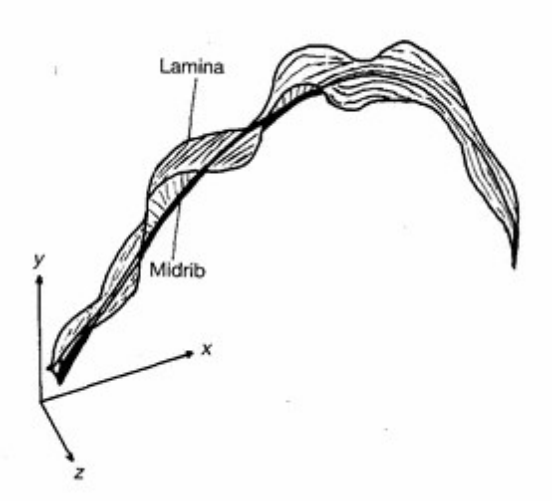

FIGURE 1.5: General morphology of the mature maize leaf [116]

the variation of curvature, and $\delta(M_z)$ is the variation of the bending moment between the initial and final equilibrium state. However, for the suppleness of an equivalent homogenous beam, classical sf can be defined by equation (1.2); where E_{eq} is the longitudinal Young's modulus of the equivalent material and I_z is the second moment of inertia around the z axis.

$$sf = \frac{\delta(C)}{\delta(M_z)} \tag{1.1}$$

$$sf = (E_{eq} I_z)^{-1} \tag{1.2}$$

Some calculation methods used in the study are curvilinear abscissa, curvature and bending moment. The curvilinear distance (s_i) is defined for each point as the sum of the upstream inter-point segments. It is defined in equation (1.3); where x_i and y_i are coordinates within the plane of symmetry of the ith point. The simple curvature (C_i) can be defined in equation (1.4); where it is calculated in each point i and P is the local fitted polynomial. Finally, the bending moment (M_{zi}) can be defined in equation (1.5); where x is the point abscissa and w is the sum of the self-weight and of eventual additional weight acting on the current point.

$$s_i = \sum_{j=1}^{i} \left[(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2 \right]^{1/2} \tag{1.3}$$

$$C_i = \frac{\left[d^2P(x_i)/dx^2\right]}{\left\{1 + \left[dP(x_i)/dx\right]^2\right\}^{2/3}} \tag{1.4}$$

$$M_{zi} = \sum_{j=i+1}^{n} \left[(x_j - x_i) \times w_j\right] \tag{1.5}$$

Some results have been concluded in the study of [116]. The overall shape of the suppleness curve is very stable, a quasi-exponential increase in suppleness along the midrib; however there is significant difference in slopes between ranks. Furthermore, the midrib represents more than 87% of the leaf rigidity within the basal quarter of the leaf, and it reaches 50% only at the middle of the leaf. Besides that, the elastic straining of the mature leaf under its self-weight explains only one-third of the leaf bending.

[42] developed a 3-d parametric model of maize (Zea mays) canopy. It was used for accurate computation of the radiative transfer, the computation for describing the vegetation functioning and interpreting remote sensing data through investigating the light climate within canopy. Measurements of dimension, shape, position, and orientation of the leaves and stems were used to give a picture of the maize canopy architecture. Here, plant structural parameters were considered, they are leaf length, leaf width, leaf height, leaf area, and diameter and height of stem. Furthermore, leaf curvature, undulation and cross section were measured at a subset of plants where the silhouette was digitized. The plant model itself was constructed in three steps: adjusting distribution laws for the parameters, drawing the values of parameters by using Monte Carlo technique, and running the algorithm that was used for radiative transfer calculation then.

To keep the implementation simple, they used a minimum set of input variables and parameters. They studied fully developed plants, however they ignored the reproductive parts and leaf senescence for simplification. Finally, they used the developed 3-d canopy architecture to compare the SAIL reflectance model [168] with PARCINOPY which is a Monte Carlo ray tracing model. The result was that PARCINOPY was confirmed as a very convenient tool to evaluate simpler reflectance models applied to specific vegetation types.

[162] developed a FSPM to calculate annual photosynthetic gains in beech saplings (Fagus crenata). The calculation of annual photosynthetic gains was technically used to determine an influence of foliar phenology and shoot inclination on the carbon economy. In detail, to estimate an hourly leaf photosynthetic gain, [162] calculated the hourly light interception of each leaf; and to evaluate the importance of simultaneous foliar phenology

and slanting shoots, they calculated the photosynthetic budgets with contrasting foliar phenology and shoot inclination.

Concerning shoot inclination and foliar phenology, self-shading is greater in upright shoots than in slanting shoots; and the self-shading can reduce the intensity of light. Without a vertical gradient in light intensity, the light intensity at the top of a shoot does not depend on shoot inclination. If leaves on a shoot open and fall simultaneously, lower leaves are more or less shaded by upper leaves throughout their lifespans, including the period when they are young and productive. One possible solution for avoiding strong self-shading during the productive period of a leaf is to open leaves successively and expose them to strong light when they are young.

To implement shoot inclination and foliar phenology, [162] used two kinds of research methods. The first method is functional-structural plant modelling. The model is composed from sub-models, such as: 3-d structure of aboveground tree parts; foliar phenology, the concept of simultaneous foliar phenology of beech [81] was used in this research; light interception, in this part they used a detailed light model [52]; instantaneous photosynthetic rate; and alternative traits. The second method is simulation. It was developed to simulate each individual model separately.

In a similar study, [60] used a FSPM for investigating the influence of geometrical traits (internode length, leaf area, branching angle and shoot top diameter) on light interception efficiency of trees (case study in apple trees). They chose the silhouette to total area ratio (STAR) of leaves for evaluating the level of light interception efficiency. The FSPM, here, means an architectural model for simulation of apple trees' topology and geometry, MAppleT [31]. The apple tree topology was organized according to Markovian models that control both the branching patterns and growth units (GU) successions along axes [31] [57] [30] [135]. In addition, regarding geometry, branch bending was simulated with a biomechanical model [131] and the secondary growth of each internode (causing diameter expansion) was simulated with the pipe model [145].

A silhouette area is a projected area of an object on a plane that is perpendicular to the projective direction (see Figure 1.6). It was used to calculate $STAR$, where $STAR$ is given in equation (1.6). In equation (1.6), PLA is total projected leaf area (the silhouette area of the tree) and TLA is the total leaf area, where TLA is given by equation (1.7), where A_i is the surface area of leaf i and n is the total number of leaves.

$$STAR = \frac{PLA}{TLA} \tag{1.6}$$

$$TLA = \sum_{i=1}^{n} A_i \tag{1.7}$$

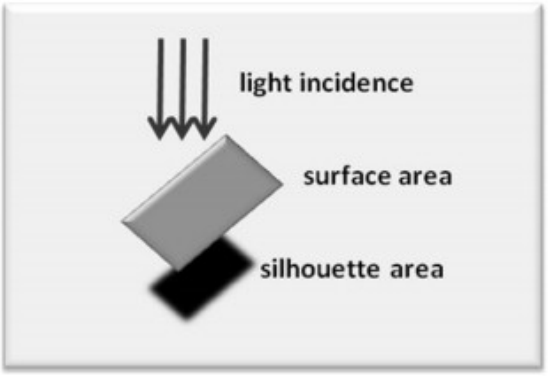

FIGURE 1.6: Illustration of silhouette area [60]

Indeed, there are three kinds of aspects that were expected to have direct influence on the STAR value: the leaf surface, it is determined by the leaf area (LA); the density of leaves, it is determined by internode length, branch bending and branching angle; and the leaf orientation, it is influenced by branching angle and branch bending. Thus, technically, the branch bending is directly determined by leaf area, internode length and shoot top diameter. Four geometrical traits: leaf area (LA), internode length (IL), shoot top diameter (TSD), and branching angle; have indirect influence on the STAR value. Figure 1.7 describes the relations between the investigated traits and putative influences on light interception efficiency.

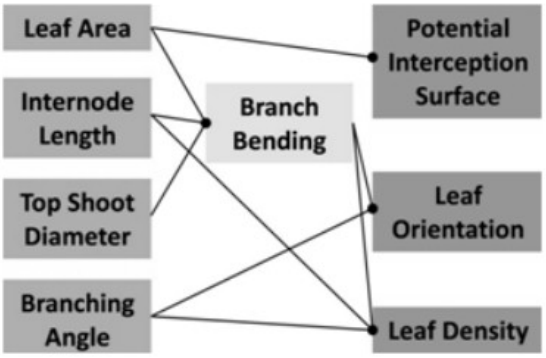

FIGURE 1.7: Relations between the investigated traits and putative influences on light interception efficiency [60]

Here, [60] simulated the growth and branching process of apple trees in MAppleT with Markov Chains and Semi Hidden Markov Chains. So, the output variance of STAR results from the stochastic part of these models. In visualizing the result, they run a

first set of 300 simulations with ordinary stochastic processes at first, and then another set of 300 simulations.

To avoid very time-consuming repetitions of simulation runs, they did not carry out a sensitivity analysis based on eFAST or Sobol methods; however they restricted their study to a set of 300 parameter combinations, in order to evaluate the stochastic part, and adopted a meta-modelling approach to investigate their model response. For that reason, they investigated two kinds of meta-modelling approaches: a multiple polynomial linear metamodel (PLMM) and a generalized additive model (GAM) (Faivre, unpublished) [179]. The PLMM aimed at modelling the simulated STAR variable as a linear combination of cross-product of polynomial functions of maximum degree between the different parameters. Here, the $STAR$ is formulated in equation (1.8). Where, the sum for j is over the (C_D^{K+D}) combinations of cross products of functions with different powers on all the $K = 4$ factors; $Z_{j,i} \prod X_{k,i}^{d_k}$ for $0 \leq d_k \leq D$, D is the maximum degree of the polynomial; θ_j are the parameters of the regression; and ε_i is the residual error term. In addition, the GAM modeled $STAR$ as an additive sum of nonparametric functions of each input parameter (equation (1.9)).

$$STAR_i = \sum_j \theta_j Z_{j,i} + \varepsilon_i \tag{1.8}$$

$$STAR_i = \sum_k f_k(X_{k,i}) + \varepsilon_i \tag{1.9}$$

Several researchers also developed virtual plants by using the softwares AMAP and GREENLAB. [33] simulated light transmission (photosynthetically active radiation or PAR) of virtual coconut stands and predicted intercrop yields by combining the results of intercropping experiments and the simulated light transmission. The virtual coconut stands are described with some parameters: trunk diameter, trunk height, trunk projection, the number of green fronds per tree, frond length (panicle and rachis), leaflet horizontal angle and leaflet vertical angle. Virtual coconut trees from AMAP were used. The results said that there is a linear relationship between PAR transmission to the ground and tree density with high coefficients of determination.

The PAR quantity incoming from each direction is calculated in two steps. The first step is a calculation of direct and diffuse components of global radiation; the calculation is based on the ratio of global radiation on extra-terrestrial radiation. The second step is an approximation of diffuse radiation within 46 sectors by combining the brightness of a clear sky formula [41] and the brightness of a standard overcast sky formula [6].

In simulation of virtual coconut stands, [33] created triangular and square designs for analyzing the effect of the planting patterns and the tree density. Other designs are created by removing some trees from the triangular design that was conducted in the Davao Research Center (DCR) of Philippines Coconut Authority; and by thinning down to 107 and 72 trees / ha. In addition, regarding a relative measurement, the main concern for characterizing the radiative conditions is to assess the quantity of transmitted PAR by determining not only the mean transmission rates, but also the distribution of light on the ground.

In addition, [136] simulated the amount of light absorbed at organ and plant scales from crop emergence to maturity by using the AMAP software. They developed a 3-d virtual sunflower, a species for which they also conducted greenhouse and field experiments (in 1998, 1999 and 2001) to parameterize the AMAPsim architectural model [9] to simulate sunflower growth and architecture, to adapt the MMR radiative calculation model [33] to greenhouse condition, and to test and simulate all experimental situations (depending on many changed variables: density, yes or no artificial shading, and yes or no decapitation) based on a combination of both models. The MMR radiative calculation model itself consists of three models, they are: Mir, it deduces the interception of directional light by vegetation elements; $Musc$, it calculates the multiple scattering of intercepted light and the resulting additional irradiation of vegetation and soil [32]; and *Radbal*, it combines outputs of previous modules in order to obtain the complete light balance of the plot for the light condition of the treatment considered [34].

In this research, the fraction of PAR intercepted by the canopy (ε_i) is calculated by equation (1.10), it is estimated from the daily incident PAR above the canopy ($PAR_{incident}$) and the daily incident PAR at the level of the soil ($PAR_{incident(soil)}$). Moreover, the daily PAR absorbed by an organ ($PAR_{a(organ)}$) is calculated by equation (1.11); where $PAR_{i(organ)}$ is the calculation of PAR absorbed by a plant organ includes its interception of incident PAR, $PAR_{s(organ)}$ is the additional fraction of light scattered by vegetation and soil which is intercepted by the organ, ρ is the reflection coefficient determined for the PAR range, and τ is the transmission coefficient determined for the PAR range. Then, the fraction of absorbed PAR (ε_a) can be calculated by equation (1.12); where $S_{(plot)}$ is the plot surface area. And, the relative leaf irradiation (RLI) is calculated by equation (1.13); where RLI is calculated as blade PAR irradiation divided by incident PAR.

$$\varepsilon_i = \frac{PAR_{incident} - PAR_{incident(soil)}}{PAR_{incident}} \tag{1.10}$$

$$PAR_{a(organ)} = (PAR_{i(organ)} + PAR_{s(organ)}) \times (1 - \rho - \tau) \tag{1.11}$$

$$\varepsilon_a = \frac{\sum PAR_{a(organ)}}{PAR_{incident} \times S_{(plot)}} \tag{1.12}$$

$$RLI = \frac{PAR_{i(leaf)} - PAR_{s(leaf)}}{PAR_{incident} \times S_{(plot)} \times LAI} \tag{1.13}$$

The results of this research said that the fraction of PAR intercepted by plants did not significantly differ from the simulated data. The fraction of absorbed PAR (ε_a) showed a sigmoidal pattern over thermal time, it was first fairly low then increased rapidly. Furthermore, changes in RLI over thermal time showed a similar pattern in all different experiment situations. And, the contribution to light absorption made by stems and petioles was minor. Finally, the simulations showed that average PAR hourly absorbed by the plant leaves, with accounting for or without accounting for heliotropic movements (especially on a day with clear sky conditions), had only minor differences [136].

Moreover, [182] designed the GREENLAB model to deliver a mathematical plant growth model that integrates plant architecture and morphology aspects with biomass production and allocation. The model produced 3-d views of the dynamics of plant growth in some aspects: leaf size, pruning, result of internal competition for resources, and the plant's phenotypic plasticity simulation. A non-linear relationship between leaf size and assimilation, and some functions that describe the general shape of each organ from its initiation to maturity; are two kinds of model functions that are included in the model.

There are several sorts of parameters used in GREENLAB, such as: topological parameters (such as physiological age and number of microstates), functioning parameters (such as base temperature, number and thermal duration of growth cycles, thermal duration of organ expansion, water use efficiency, etc.), state variables (such as number and size of organs) and geometric and allometric parameters (such as insertion angles of organs, specific leaf area and organ water content). One species as an object of the research used by [182] to adapt the model was the sunflower plant.

In addition, one application of GREENLAB is modelling of biomass. A non-linear function $f(S,\ r_1,\ r_2)$ in equation (1.14) describes the biomass production; where it depends on leaf surface area (S), and r_1 and r_2 are parameters that have to be computed based on [184] optimization methods. The biomass production of each leaf becomes proportional to leaf surface if the parameter r_2 is zero [182]. The equation (1.15) shows

that the biomass Q_n is generated by all green leaves; where N_k^L is the number of green leaves, S_k is the leaf surface area, and k is a leaf rank.

The biomass increment of organ O ($\triangle q_{(i,n)}^O$) can be described in equation (1.16); where $P_O(i)$ is the reference sink value, Q_{n-1} is the biomass supply provided by the leaves for distribution among sinks present in the plant architecture that constitute the overall demand (D_n). In addition, the sink ($P_O(i)$) can be calculated by equation (1.17); where t_O is growing organ period, a and b are real numbers calculated by heuristic methods [184]. The D_n, as the bulk demand of all growing organs during period t_O, can be calculated by equation (1.18); where O is the organ type corresponding to the leaf blade, peduncle, internode, secondary growth rings on the stem and fruits or flowers (they are denoted as B, P, I, C and F respectively). Furthermore, the total biomass ($q_{(i,n)}^O$) in the organ is the sum of $\triangle q_{(i,j)}^O$ (where j is the growth cycle), it can be calculated by equation (1.19).

$$f(S, r_1, r_2) = \frac{1}{r_1/S + r_2} \tag{1.14}$$

$$Q_n = \sum_{k=1}^{n} N_k^L f(S_k, r_1, r_2) \tag{1.15}$$

$$\triangle q_{i,n}^O = \frac{P_O(i) Q_{n-1}}{D_n} \tag{1.16}$$

$$P_O(i) = \left(i + \frac{0.5}{t_O}\right)^{a-1} \left(1 - \left(i + \frac{0.5}{t_O}\right)\right)^{b-1} \left(\frac{1}{t_O}\right) \tag{1.17}$$

$$D_n = \sum_{B,P,I,C,F} \left(\sum_{i=1}^{t_O} P_O(i) N_{n-i+1}^O \right) \tag{1.18}$$

$$q_{i,n}^O = \sum_{j=1}^{n} \triangle q_{i,j}^O \tag{1.19}$$

The competition from inflorescences makes the biomass production increase only until flowering and decrease thereafter. This condition affects the biomass partitioning, where its partitions to leaves and internodes increase until flowering. This is in accordance with classic results, where exponential growth is followed by linear growth.

Still in GREENLAB research, [150] used the architectural mathematical model GREENLAB to assist plant phenotyping (involving an organogenesis mutant) on two rice genotypes: Nippon Bare (NB) and Phyllo. In their experiment, seedlings of the wild type NB and one of its T-DNA organogenesis deficient mutants named Phyllo were investigated to analyze the variability in carbon sink to source processes among genotypes; the plant development (leaf and tiller appearance) was monitored to estimate the phyllochron and tillering rate; an Li-6200 gas exchange analysis system was used to measure assimilation rate. Moreover, the topology was characterized in terms of leaf position on tillers and the main stem, and tiller position according to the Katayama scheme [61]; and sugar content analysis was done according to [102].

GREENLAB fittings on observations were good except for NB, mainly because of the difficulty of correctly optimizing numerous small leaf sinks (generated by the growth process) compared to a single big root sink compartment. Furthermore, GREENLAB successfully sensed the phenotypic differences underlined experimentally, in particular, modified carbon sink to source relationships for Phyllo. In addition, modelling analysis of rice tillering cases shows that GREENLAB has a great potential as heuristic approach to be applied to plant phenotyping.

GREENLAB, as a method, was also used by [96] in conducting their research to build a breeding strategy simulation for predicting and improving phenotype traits. Three methods used by [96] are: GREENLAB, Quantitative Trait Loci (QTL) Cartographer, and genetic algorithm. The GREENLAB model formalism was used as a dynamic model taking into account architectural plasticity of the plant and biomass allocation at organ level. In this case, the model was used to link parameters of the plant growth model to QTL. For that reason, virtual genes and virtual chromosomes were defined to construct a simple genetic model that drove the settings of the species-specific parameters of the model. Thus, QTL detection in simulated plant traits was studied by using the QTL Cartographer software. It illustrated the chain from genotype to phenotype with virtual data that allow a simplification to make plant modelers more familiar with the benefits of growth models for breeding work. Finally, the identification of yield maximization based on the model parameters and the associated allelic combination was done by implementing a genetic algorithm method. It was computed to find the parameters, and therefore the associated genotype, that give the best yield under constant environment parameters.

The configuration of the GREENLAB model is illustrated in Figure 1.8. It is a generic growth model based on dynamic equations that integrate organogenesis, biomass allocation and production at the organ scale [96]. Here, the genome of the plant influences the endogenous parameters and the rules processing the environmental impacts. Thus,

the endogenous parameters are converted to the mathematical growth model to produce virtual plant growth. In the mathematical growth model, the various feedbacks among organogenesis, biomass allocation and production are represented by circular arrows.

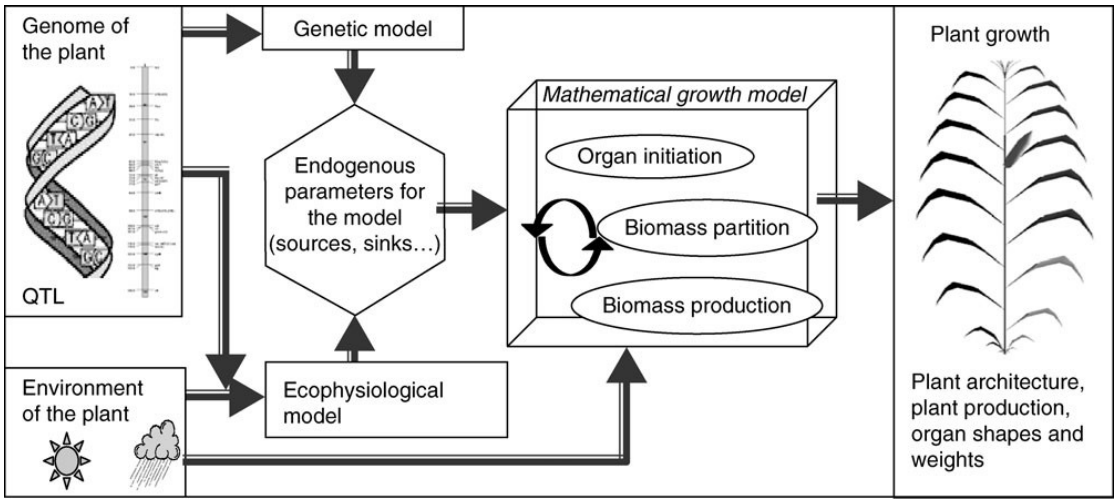

FIGURE 1.8: General flowchart linking the genetic model to the GREENLAB model [96]

A genetic model part (see Figure 1.9) was constructed to introduce a plant genotype into the growth model; and some parameters were chosen to be considered. Those parameters for simulation of this model were taken from the calibration results of [59] and [103] on Zea mays L. The endogenous parameters themselves were distinguished on the basis of the stability study made by [103]. Those parameters were identified in an array Y with size T; where T is the number of genetic parameters.

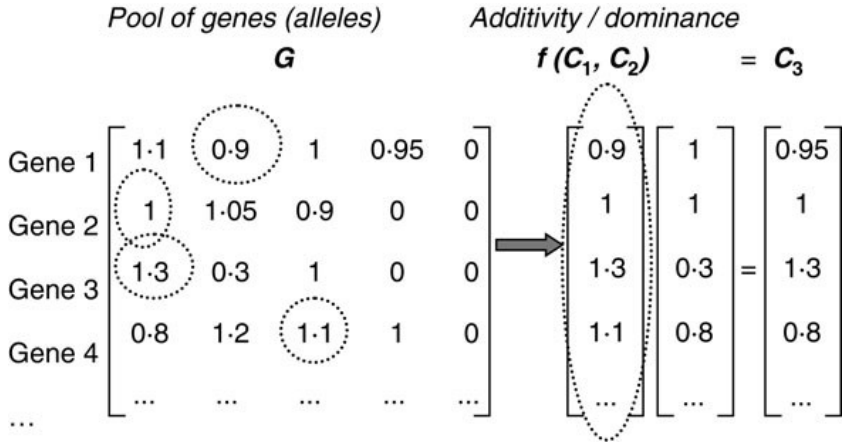

FIGURE 1.9: Genes and alleles expression [96]

Furthermore, the virtual genome of the plant is assumed to consist of only one pair of chromosomes. Genes (located on the chromosome) can be assumed to be numbers (quantitative); and each gene can take several values called alleles. They are represented

in the matrix G (N x P); where N is the number of genes and P is the current maximal number of alleles for one gene (see Figure 1.9). The matrix is used to build the genotype of the plant. In addition, the chromosome C (C_1 and C_2) is a vector of size N whose components are taken from the matrix G (one allele in each line). Moreover, C_3 (with size N) is the fictitious chromosome of allele effects; it is resulted from the application $f(C_1, C_2)$ as a set of rules for each component of the chromosome vectors. From the virtual chromosome C_3, the genetic vector of parameters is calculated as a product of matrices $Y = DAC_3$; where Y is the array of the parameters to set, D is a diagonal matrix (T x T) whose coefficients are scaling factors to have range compatibility, A is a (T x N) matrix defining the influence of genes on each parameter, and T is the number of genetic parameters.

The genetic algorithm results gave the allelic combination that optimizes cob weight under given environmental conditions. Blade thickness and blade resistance needed to be as small as possible as their diminution increases the plant's ability to perform photosynthesis. On the other hand, large seed biomass gave a stronger plant. Sinks of unproductive organs (except cob) should take minimal values to avoid waste in biomass partitioning. The number of short internodes should be as large as possible as it lets the plant allocate biomass uppermost to the blades that are the future sources of assimilate production. However, the optimization results found for cob sink and the cob sink variation parameters are coherent with the observations that tend to show the existence of an optimum point not situated on the interval boundaries. The increase in cob weight induced by the parameter optimization is about 60%.

[68] also used the GREENLAB model in their research. They used it to simulate the growth of a tree. This research was conducted to find a tree shape maximing its light interception. In the tree's light interception, there are three important aspects, they are: phyllotaxy, branching, and bending. Phyllotaxy is the arrangement of leaves and branches on a plant stem. To describe the phyllotaxy, GREENLAB uses a parameter Φ (0^o to 360^o) which is defined as rotation angle between two adjacent internodes. Concerning branching, the branching angle is the angle between a branch and its mother stem; it is defined in GREENLAB as parameter θ (0^o to 180^o). Bending depends on gravity and phototropism. The gravity will bend branches downward and the phototropism will bend branches upward. There are two parameters K (larger than zero) and p (ranging from 0.0 to 1.0) that control the degree of bending and the position where the branch begins to fold upward.

In addition, to compute the light interception, [68] summed up the light interception from individual organs. Since a leaf is represented as a mesh object in computer memory, several rays were emitted evenly into the sky sphere from each point of the leaf mesh.

For each ray, they counted leaves that this ray encounters. Let this number be n. The *visibility* of this ray is calculated as t^n, where t is the light transmittance of a leaf. Finally, the *visibility* of a leaf is estimated using the mean value of visibility of all rays emitted from this leaf mesh. The authors use the sum of *visibility* of all leaves (V) to measure the light interception of the tree.

The visibility of all leaves (V) can be written as a function of four parameters, $V = f(\Phi, \theta, K, p)$. It is a typical optimization problem to find a set of parameters to maximize V. For this purpose, [68] used the Particle Swarm Optimization (PSO) algorithm [144].

Several researchers have conducted their research in virtual plants by using another type of model, such as LIGNUM. The model LIGNUM was developed by [125], it describes the three dimensional structure of the tree crown (segments, branching points and buds) and defines the growth in terms of the metabolism taking place in units which correspond to the organs of the tree. Each pair of tree segments is separated by a branching point; and the buds produce new tree segments, branching points and other buds. The tree segment itself consists of sapwood, heartwood, bark and foliage. In this model, the architectural structure of the tree is formed by simple branching rules. In addition, the main emphasis in the LIGNUM model is on the carbon balance formulations, and the central question is how to incorporate the carbon balance in a model tree that consists of a large number of units.

As a basic assumption, the tree growth model implemented by LIGNUM has some main characteristics: the carbon balance is considered on an annual basis; photosynthesis, respiration, senescence and growth are included in the carbon budget of the tree that can drive tree growth; the allocation of growth to new and existing parts of the tree is modeled at the tree level; the pipe model hypothesis and the principle of functional balance are taken into account [118]. According to the pipe model hypothesis, a tree can be pictured as consisting of foliage, fine roots and a bundle of pipes [164]. The active pipes extend from the root tips to the foliage elements, and the disused pipes no longer connect the roots and the foliage elements. A cross-sectional area of the active pipes corresponds to the cross-sectional area of sapwood. Accordingly, the heartwood consists of disused pipes. The original pipe model idea was further modified to allow for the observed dynamics of the active pipes that the dying foliage releases for reuse [118].

[125] assumed that the foliage biomass is associated with a certain cross-sectional area of sapwood below the foliage. Thus, the new wood growth is proportional to the net change of foliage above the tree segment in question. It means that the cross-sectional area of sapwood in a tree segment just below a branching point is equal to the sum of the cross-sectional areas of sapwood in the tree segments going upwards from that branching point. They also assumed that the foliage density on the surface of a new

tree segment is constant. This leads to a linear relationship between the cross-sectional area of the sapwood and the foliage mass in a new tree segment. The functional balance hypothesis states that in cases where the soil conditions remain constant, the amount of foliage and roots in a tree are linearly related [118].

In addition, [99] adapted the LIGNUM model to model trees by combining tree metabolism with a realistic spatial distribution of morphological parts; the model has been calibrated for Jack pine. A sensitivity study indicated that uncertainty in the photosynthesis and respiration parameters will primarily cause changes to the net annual carbon gain, which can be corrected through calibration of the growth rate; it means that the effect of uncertainties and errors in the model parameters can be minimized through the calibration process. Furthermore, the model can simulate the effect of a decrease in light level on height, biomass, total tree branch length, and productivity; and the results were compared with field data.

1.3.2 Rewriting Systems

1.3.2.1 L-Systems

In 1968, the biologist Aristid Lindenmayer invented a special sort of grammar to describe the growth of arrangements of plant cells [95]. This approach was called Lindenmayer systems (L-systems for short) later on. In the mathematical theory of plant development, the central concept of L-systems is rewriting. The rewriting is a technique for defining a complex object by successively replacing parts of a simple initial object (ω) using a set of rewriting rules or productions (p). The rewriting concept in L-systems is different from the rewriting concept in classical formal grammars (Chomsky grammars of [29]), especially in the production part. In Chomsky grammars, productions are applied sequentially; whereas in L-systems, they are applied in parallel and simultaneously replace all letters in a given word [132].

L-systems that can be used to construct developmental models of herbaceous plants (non-woody plants) are specified at three levels of detail: partial L-systems (abstract level), L-systems schemata (control mechanism level) and complete L-systems [132]. The partial L-systems employ the notations of nondeterministic L-systems to define the realm of possibilities within which structures of a given type may develop. The partial L-systems capture the main traits characterizing structural types, and provide a formal basis for their classification. In this level, there are five mechanisms: stochastic, table L-systems, delay, accumulation of components, and signal mechanism. Furthermore, L-systems schemata is a level used to describe the topology of individual plants and

temporal aspects of their development. Finally, the geometric aspects (including information concerning growth rates of internodes, the values of branching angles, and the appearance of organs) are added at the complete L-systems level.

The L-systems can be used in a broad area of research, like what [73] did. He tried to construct a particular (biological) species development by using L-systems. In the construction, there are eight steps which should be followed, they are: analysis of the biological object, definition of informal rules, definition of L-systems axiom and rules, computer simulation and interpretation of generated strings, translation into a graphical output, comparison of the artificial object with the behavior of the real object (in the model development concept, it could be called a validation process), and correction of the L-systems and repetition of the steps before (if necessary; in the model development concept, it could be called a verification process). Regarding an approach for parameter-fitting, Genetic programming (GP), the L-systems also can benefit. Basically, GP is a method to automatically develop populations of computer programs through simulated evolution [87] [88]. The combination of L-systems and GP will be called genetic L-systems programming. Genetic L-systems programming is conceptually similar to the GP paradigm introduced by [87]; however one main difference between the two concepts here is the use of higher-order building blocks (they are called 'patterns') for generation and modification of expressions, where basically each expression is constructed from a start pattern in a recursive manner by combining some expressions that are taken from the expression pattern pool (see Figure 1.10).

Technically, the population of genotypes in an L-systems consists of symbolic expressions (data structures). The head symbols denote (abstract) data types for which decoding, interpretation and evaluation functions are easily definable by mechanisms of pattern matching. Regarding the fitness concept, it could be derived from the L-systems interpretation functions so that each L-systems genotype technically gets an associated fitness value. Furthermore, to develop the next generation of expressions, a genetic operator op_i is selected from an 'operator pool' OpPool $= \{op_1, \ldots, op_k\}$ depending on its operator rank $r(op_i)$. Each operator op_i performs a mapping $o(op_i)$: $G^{a_1(i)} \rightarrow G^{a_2(i)}$ from an $a_1(i)$ to an $a_2(i)$ dimensional genotype vector where G represents the set of genotype expressions. In addition, the size and shape of the expressions change dynamically during the evolution process through genetic operators [73].

To perform pattern mutation on an individual expression (Figure 1.10 (a)), a mutation template is selected according to the pattern ranks (see Figure 1.10). Suppose the first template has been selected with demanding three arguments for the *Stack*. The sub-expression with head *Axiom* is replace then by a newly generated *Axiom* with *Stack* argument expression in a modified individual genotype (Figure 1.10 (b)). On the other

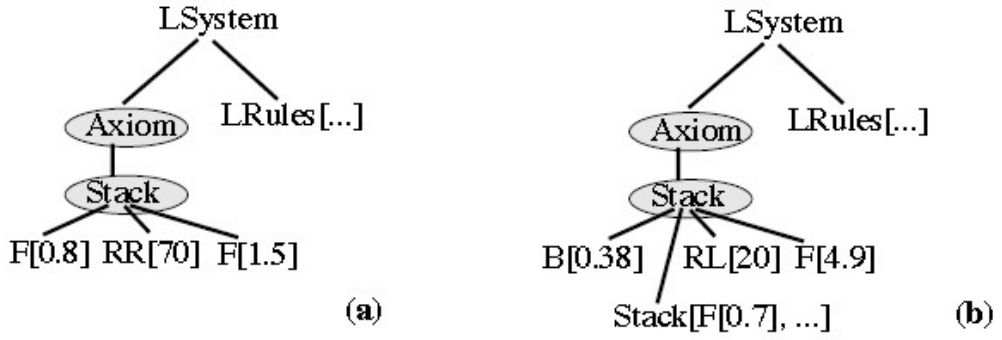

FIGURE 1.10: Pattern mutation on an L-systems genotype [73]

hand, pattern crossover is used as a recombination operator which enables a structure exchange of the same type between two similar individuals (Figure 1.11).

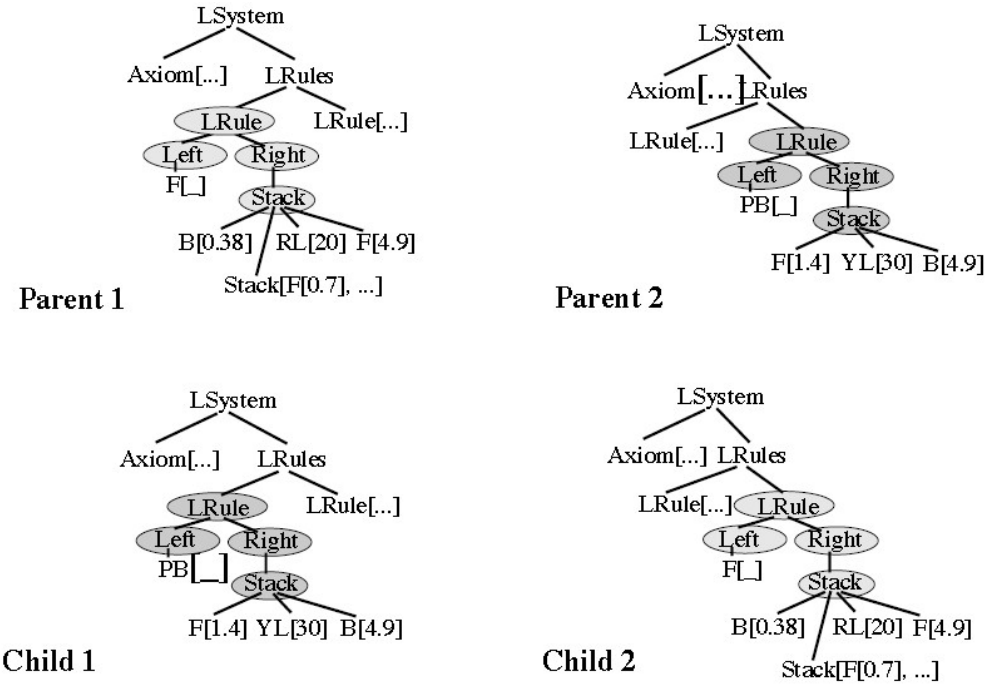

FIGURE 1.11: Expression recombination: pattern crossover [73]

L-systems are a fruitful basis for research in virtual plant evolution. Already one year later, [74] continued his work on GP by using L-systems. He described genetic L-system programming by using Mathematica for developing a model of breeding and evolving artificial flowers. In L-systems, each program encoded as a symbolic expression technically has to be interpreted, and an assigned fitness value is dependent on the optimization task to be solved. There are four expressions commonly used in Genetic L-systems in developing models of genetics: mutation, it replaces a randomly selected sub-expression by an expression with the same head generated from the expression pool; crossover, it

is a recombination operator between two or more expressions, sub-expressions with the same head are selected within the expressions and interchanged; deletion, it erases expression arguments whenever this is possible according to restrictions of the number of arguments for the selected expression; and permutation, it interchanges the arguments sequence of an expression [74].

Let us now consider some applications of the L-system approach an agricultural crop plants. In 1999, [49] successfully developed ADEL-maize. It is an L-systems based crop simulation model obtained by combining a 3-d model of maize development with a physical model computing the microclimate on the 3-d structure. It was implemented to scientifically simulate the development of maize depending on organ temperature. Numerous variables are used here, for instance: the time step, the temperature, the growth rate, etc. The model was complemented (from a previous version) by considering dry matter availability; where dry matter production is calculated from the light intercepted by individual plants. ADEL-maize also includes a model of carbon allocation and rules for the regulation of morphogenesis by carbon availability.

Indeed, the model consists of four components [49]. The first is a model component for growth and development of individual organs. The component represents the growth and development that are based on temperature and carbon availability; involving the processes of phytomer (one internode and the attached leaf) initiation, leaf growth, and internode growth. The component currently focuses on the aerial vegetative structures only, without considering the reproductive organs.

Concerning individual organ development, in this work, [49] presented one unique characteristic of grass leaf elongation. In their model, the elongation consists of three main stages: exponential growth during the establishment of the growing zone, linear growth and slowing down. Furthermore, the leaf elongation rate is considered to be independent of leaf rank. Its response to temperature is approximated as a linear increase between a base temperature and an optimum temperature. The elongation is modeled as a linear process with three parameters (cf. [127]): the thermal time of the onset of elongation (^oCd), the rate ($cm/^oCd$), and the duration (^oCd). The elongation duration (including internode elongation duration) is essentially calculated from the growth rate and the final length. In addition, leaf shape is developed on a plane surface and is determined by the relation between width and distance to leaf tip [128] in [127].

The second component is a model component for production and allocation of dry matter. In this case, the component is used to calculate a carbon availability to individual organs. And the conceptual scheme of interactions between temperature and carbon availability here is based on the classical crop model CERES-maize [79]. In addition, at

each time step, the growth of organs is practically determined as a function of temperature, and then converted to dry mass requirement based on a minimal value for specific leaf area and specific internode volume. When the total dry mass production exceeds the needs of all growing organs, the dry mass is stored. The stored dry mass is not available for growth during the vegetative stage.

The third is a model component of organ shape that enables a 3-d representation of the plants. The representation is performed through a process of geometric interpretation of parameters relative to the shape and the organs. The geometric interpretation uses standard facilities incorporated in the L-systems based software 'Graphtal' [153], and requires a geometric parameterization of each module.

The last component is a model component that is used to calculate the distribution of light and temperature at specific sites in the 3-d structure of the plants. It can also compute the interception of light by individual plants (PAR); where the PAR is required to estimate the dry matter production. It also calculates the temperature of the growing zone. In addition, the model as well can calculate sun and sky direct light by projecting the 3-d structure along a set of directions sampling the sky hemisphere. For calculating the temperature regime, an energy balance model of [25] was implemented. All calculations are done at each time step for each plant.

Concerning the light interception, three levels of density resulted; those densities significantly affect the fraction of total dry matter that corresponds to the structural needs of the leaves. At the lowest density, the radiation interception increased with increasing leaf area throughout the period of simulation. Each new leaf element contributed to an augmentation of dry matter production. For two other densities, the intercepted PAR rapidly reached an asymptote, creating a constant regime for resource acquisition, independently of further development of the aerial structures.

Furthermore, [127] technically used the L-systems based ADEL-maize model [49] to realize virtual plots with the same given arrangement of individual plants. The model was able to simulate 3-d canopies and employs a projection method that was used to calculate the radiation interception. Two types of simulation were performed by taking account (or not) of the adaptation in leaf size that was observed in the field, especially for plants neighboring gaps. The simulation was able to estimate the intrinsic role of plasticity in leaf dimension for light interception and the consequences for grain yield. It is important to investigate the genetic basis of plasticity, because some genotypes seem to compensate better than others. In this research, the field experiment itself was carried out in 1995 at Grignon (Yvelines France), with the Déa hybrid of maize (Zea mays L.) as the research object.

The research idea indeed came from the question about the plants neighboring gaps in the stand. Usually, the plants at gaps have a better grain yield, because they intercept a larger amount of light. However, which part of this increase is due to larger incident radiation and which part to change in plant morphology is hard to find out in the field.

From the result of the light interception calculation of individual plants, consequently the model is able to estimate the grain production of these plants. This estimation is done based on experimental relations between intercepted light and yield. In addition, the consequences at the yield level are evaluated. This evaluation is done by comparing the calculated light interception and yield for simulated 3-d canopies with irregular plant spacing and for simulated 3-d uniform canopies having the same mean population density.

The result said that the planting density of the stand modifies the radiation available to plants. It means, the radiation available for plants located inside the canopy differs from radiation available for plants neighboring a gap. And the model showed the possible effects of changes in available radiation on leaf size. Actually, an increase in leaf width with light availability is a general feature in grass plants.

Another result was that from emergence to flowering, plants bordering gaps intercept more light and that radiation follows the ranking according to the widths of the upper leaves, both depending on quantity of light available. In addition, the intercepted radiation in an irregularly spaced plant canopy is less likely to be converted to grain, because it reaches lower parts of foliage for plants bordering a gap.

In analogy to their maize model, [50] successfully constructed the L-systems based model ADEL-wheat. It is a 3-d architecture model of wheat development. The model simulates the kinetics of elongation and geometric shape of individual organs, based on major traits of vegetative development of wheat plants. Two field experiments were organized and done at INRA Grignon – Paris, by using two kinds of winter wheat (*c.v.Soisson*) as research objects. Technically, the model was implemented by using the L-systems formalism and software Graphtal [153] that are respectively used as programming language and model platform software.

In modelling, the L-systems formalism indeed represents the plant as a string of symbols, some representing the plant organs (called modules, such as leaves, internodes, etc.), and others coding for topological or geometrical links between modules. It also symbolizes the plant development by using a set of substitution rules (the productions) that are applied simultaneously to the whole string. The plant development is described as the repetition of three sequences of events: initiation of phytomers by the apex; phytomer

development showing the succession of rapid elongation stages of blade, sheath and internode; and tiller production.

[31] also made use of the L-systems concept. They successfully addressed two required important aspects in architectural simulation, topology and geometry aspects. These aspects have been brought into a single simulation environment by using an L-systems based simulation model, MAppleT. Architectural studies, especially in horticulture, are very important. In horticulture, plant architecture plays a vital role in 3-d foliage distribution that has high consequence in light interception and also carbon acquisition, whereby it strongly affects the productive growth of the plant.

Two kinds of method, stochastic and mechanistic, were used to describe tree topology and geometry respectively. Two individuals of 1 – 6 years old apple trees of cultivar Fuji were used as the objects of the research. Several topology aspects were successfully modeled; such as the succession of growth units (GUs) along axes and the branching structure of GUs (based on hierarchical hidden Markov modelling; [45]), with implementing a four-state Markov chain and hidden semi-Markov chains (HSMCs). In the succession of GUs along axes, two parts were modeled: the first GU occurring in the axis (with initial probability parameter) and the succession of GUs along axes itself (with transition probability parameter). In the branching structure of GUs, four parts were modeled: which branching zone is the first one in a GU (with initial probability parameter), the succession of branching zones along a GU (with transition probability parameter), the lengths of branching zones in number of metamers (with occupancy distribution parameter), and the branching type composition of branching zones (with observation distribution parameters).

The shape of each branch was calculated based on a biomechanical component [51]. By simulating branch bending and twisting, the branch shape could be obtained. Technically, the bending and twisting was expressed referring to three orthogonal unit-length vectors: heading, upwards and left directions (HLU frame). The shape of the axis itself depends on the torques acting on its nodes. In calculating these torques, the gravity force and the effect of phototropism and gravitropism factors were considered. These factors also powerfully affect branch bending.

Several limitations and possible improvements of this model can be noticed. [171] mentioned the biological aspect that is rarely considered in the model. [31] as well declared that the strategy to generate branching sequences by using HSMCs is not optimum, there is a bias introduction in the branching sequence case. The improvement that should be taken in the future is by proposing a more sophisticated parameterization about tied parameters between HSMCs and covariates that influence specific parameters.

The simulation of gravities was only partly included. In the future, the purely stochastic description of branching should be augmented. Also there is an incorrectness of the solution of branch inclination for some orders, perhaps because of the use of a constant value for the pipe model component (independent of the shoot type and branching order). It is probably better if the researchers implemented the model of [39], a model refining the pipe model; or the L-Peach model of carbon allocation [101].

1.3.2.2 Relational Growth Grammars (RGGs)

Concerning L-systems, several limitations were found. particularly, the used data structure is just a linear string of symbols [84] and the L-systems are not an appropriate tool for creating properly 2-d or 3-d models [95]; for these reasons, relational growth grammars (RGGs) appeared. RGGs change the parallel rewriting system to graphs, these are structures consisting of nodes and edges [95] [64]. In the framework of plant modelling, the RGGs concept is used to model architecture, processes, and interactions during plant growth. The programming language XL implements the RGG formalism. It is a programming language that combines the rule-based programming approach of L-systems and RGG and object-oriented programming approach of the language Java. The programming language XL can be considered as a concrete implementation of RGGs for the specific needs of plant modelling [85].

1.3.2.3 The Programming Language XL and the Platform GroIMP

The programming language XL, the abbreviation of eXtended L-systems language to reminisce the L-systems concept, is an extension of the programming language Java. The programming language Java was conceptually chosen as a basic programming language to extend, due to several reasons: a wide usage, support of object oriented programming, a simple language, the widest range of libraries, and the fact that Java virtual machine compiling is more flexible than the typical native machine code compiling. In addition, as the programming language Java has no graph or rule-based features, a support of RGGs is required; thus in the programming language XL, the concept of RGGs is feasibly easier to be implemented [83]. Here, in implementing RGGs, the high-level concepts of the programming language Java are still adopted; such as the concepts of classes, methods, variable definitions, control and condition statement expressions [23].

Moreover, for use particularly in plant modelling, the programming language XL is incorporated in the modelling platform GroIMP [23]. The modelling platform GroIMP, the acronym of Growth-grammar related Interactive Modelling Platform, technically is an open-source software implemented in Java. It is licensed under the terms of the

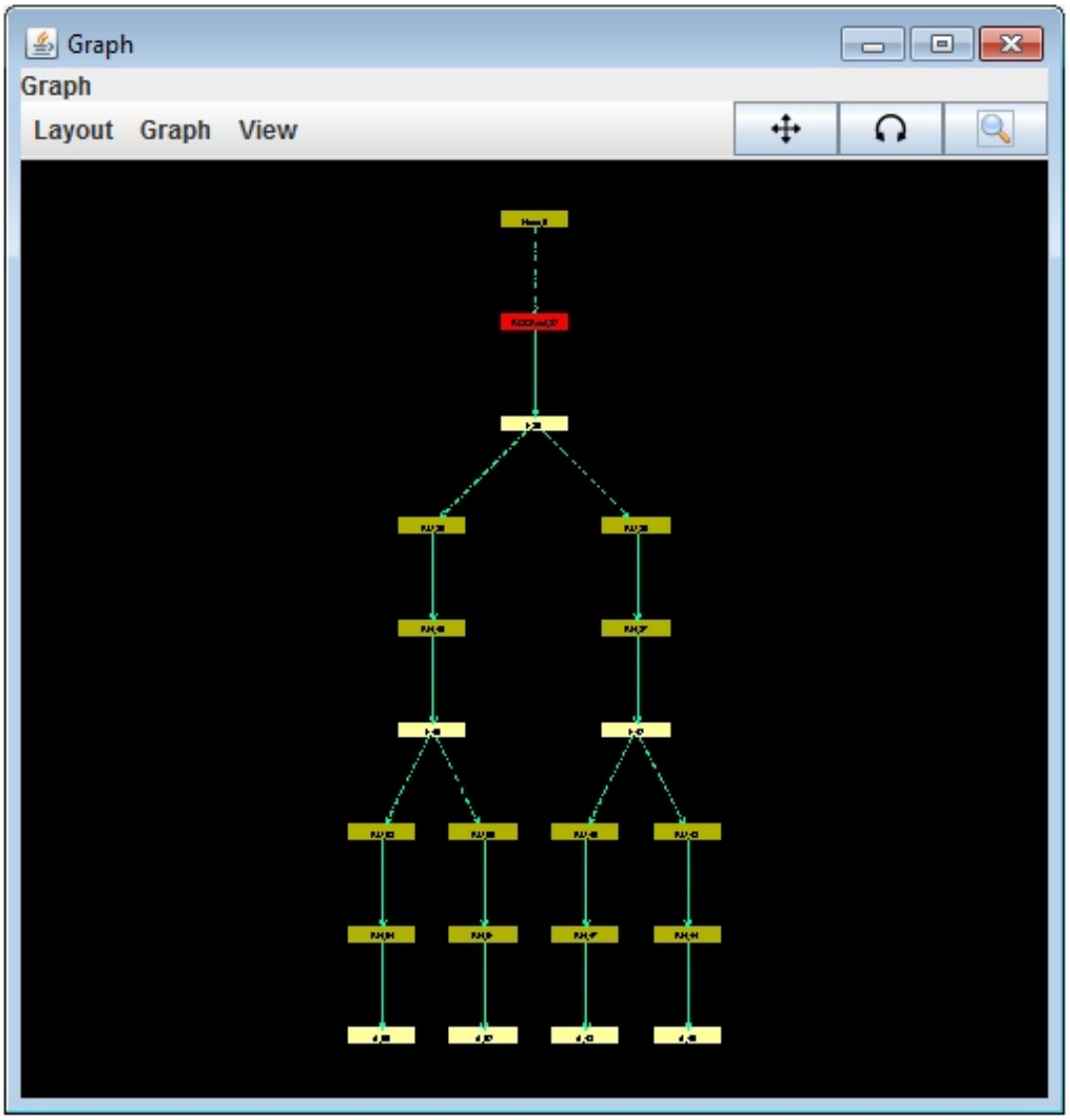

FIGURE 1.12: Graph structure in the 2d graph panel of GroIMP [86]

General Public License, version 3. The latest information and version can be found at http://www.grogra.de/. Fundamentally, based on RGGs, the modelling platform GroIMP is designed as an integrated platform [85] [83]. It incorporates modelling, visualization and interaction. Several features belonging to GroIMP make it very appropriate for FSPM, for instance: an integrated modelling backbone, a complete set of 3d-geometric classes for visualization, a visual presentation interface that allows users to dynamically interact with the model, also a light model based on the built-in raytracer that computes the distribution of light within a scene, and a visualization of the graph structure in a 2d graph panel (Figure 1.12) to represent the structure of the model that is being constructed [86], by using a set of layout algorithms that can be applied to arrange the nodes in a 2-d graph view of the data structure representing the current

model [83]. An overview of the GroIMP environment itself is shown by Figure 1.13, with easily understood part specification, such as menu bar, control part, 3d view part for viewing the model in 3d view, etc.

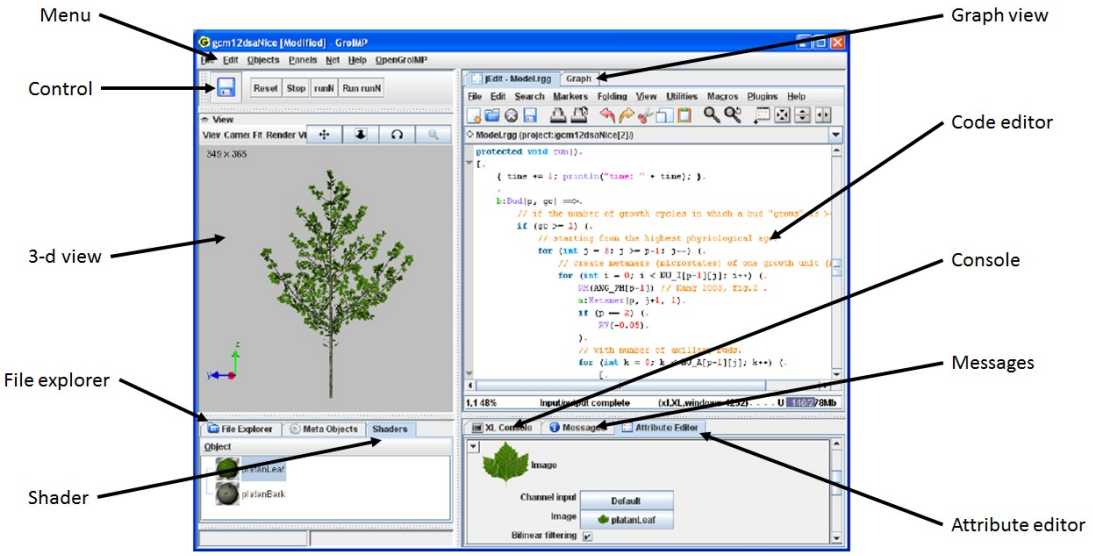

FIGURE 1.13: GroIMP environment screenshoot [65]

1.4 Rice Plant

1.4.1 Introduction to Rice Plant

Rice plays an important role in providing food to the majority of world population. It is one of the most popular foods in the world [108]. Mainly in Asia, Africa, and South America; rice has become one of the most important food crops for more than a half of world's population [173]. Concerning plant architecture, rice plants morphologically and physiologically change year by year. Several changes; such as a wider leaf area, a longer and thicker leaf, increases in leaf and tiller number, and shorter plant age [27] in [104]; surely were preferred by breeders of rice plants to keep rice existing and living in the high weather and global changes. In fact, the rice plants are very adaptable to their environment. They can grow in many different places in the word and under a variety of climates [183].

The growth period of the rice plant ordinarily can be divided into three big stages: vegetative, reproductive, and ripening stages. The vegetative stage is characterized by active tillering, gradual increase in plant height, and leaf emergence at regular intervals. The productive stage is divided into sub-periods: pre-heading and post-heading periods;

however, it is normally indicated by culm elongation (which increases plant height), decline in tiller number, emergence of the flag leaf (the last leaf), booting, heading, and flowering. And the ripening stage is typified by leaf senescence and grain growth (increases in grain size and weight, and change in grain color) [183]. Due to the rice plant's uniqueness, and particularly as so many parameters are involved in determining its architecture and development, there is a strong motivation to take it as an object for modelling and simulation.

1.4.2 Rice Plant Models

1.4.2.1 General Purpose Models of Rice Plant

The research in rice plant modelling has been being popular since several years ago. In 1959, [160] has already conducted research related to modelling rice. He investigated the relation between the rice plant's yielding ability with the assimilation system. Theoretically, the rate of dry matter production depends on the arragement of the assimilation system. This encouraged [160] to identify a relationship between the habit of the assimilation-system and the growth of crop plants. Three plant species: sweet potato, soybean and rice; were used as objects in the research. And, three attributes of a single leaf affecting the assimilation-system analyzed in the research were form, direction and arrangement.

Regarding a leaf form attribute, especially the ratio of leaf area and leaf weight (here noted as AW ratio), the varieties that have lower optimum levels of fertilization will have a larger leaf area per unit weight of leaf. That happened in almost all varieties analyzed (6 varieties of rice, 10 varieties of sweet potato, and 12 varieties of soybean). Especially in rice plants case, the AW ratio decreases linearly with increasing leaf weight (at the tillering stage; [176]). It means, the higher AW ratio has a close relation with the high yielding ability under light manuring condition. Adaptation to fertilization is one of the main factors affecting the AW ratio. In heavy manuring case, the higher AW ratio may merely increase the degree of crowding of leaves to excess and may not be so favorable to growth and yield as in the case of light manuring. In the heavy manuring case, it is more profitable to have 'thick' leaves. In addition, soybean and rice varieties adapted to heavy manuring tend to have smaller-sized leaves than those adapted to light manuring, however this condition is not marked in sweet potato varieties.

Furthermore, leaf direction was investigated. The differences in this attribute are remarkable only in rice plants. The leaf position in rice plants adapted to light manuring clearly tends to be horizontal in comparison with plants suited for heavy manuring.

It means, the thinner and horizontal leaves are suitable to receive a large amount of incident solar radiation [19].

The last considered attribute is the leaf arrangement. It was investigated by quantitatively weighing the leaf-blades at every horizontal and vertical distance from the base of the plants. It was concluded that the rice and sweet potato varieties adapted to light manuring showed 'dispersing-type' leaf arrangement for the single plant, and those adapted to heavy manuring showed 'gathering-type' leaf arrangement for the single plant. Here, the 'dispersing-type' leaf arrangement enables that all leaves of the plant can receive full light at least at early stages of growth. However, the 'gathering-type' leaf arrangement has more benefits, because: it is better organized in individual plants than between plants, leaves adapted to mutual shading can be adopted, the stem and the like-natured organs (petiole, leaf sheath, and midrib) can be reduced in quantity, and a 'ravine' of assimilation-system can be formed between plants, and this may be effective for equal distribution of light and gas among the crowded leaves.

The conclusion of this research said that there are two types of assimilation-system that can be formulated: the assimilation-system of the typical varieties adapted to heavy manuring, by implementing the combination of 'thicker and smaller' leaves, near vertical position of leaves and 'gathering-type' leaf arrangement; and the assimilation-system of the typical varieties adapted to light manuring, by implementing the combination of 'thinner and larger' leaves, near horizontal position of leaves and 'dispersing-type' leaf arrangement.

In an approach to model another apsect of the rice plant, [54] successfully developed a rice clock model. It is a computer simulation model that can indicate the processes of rice development day by day (a clock itself indicates the time process). Based on the concept of crop development that involves the processes of crop phenology, physiological age and appearances of various morphological organs such as leaf blades, leaf sheaths, tillers, roots, stem internodes and panicles; it consists of four sub-models: phenological, physiological leaf age, total leaf number and organ development.

The basic mathematical model of the phenological sub-model is described in equation (1.20) and then can be simplified into equations (1.21) and (1.22), with assumptions: $T = T_L$ for $T < T_L$, $T = T_U$ for $T > T_U$, $D = D'$ for $D < D'$, and $\left(\frac{T-T_L}{T_O-T_L}\right)^P \left(\frac{T_U-T}{T_U-T_O}\right)^Q = 1$ for $\left(\frac{T-T_L}{T_O-T_L}\right)^P \left(\frac{T_U-T}{T_U-T_O}\right)^Q > 1$. Where M is the development process for a given phase ($0 \leq M \leq 1$); N is the number of days covering the phase; dM/dt is the development rate within the phase; T is the mean temperature during the phase; T_U is the upper temperature limit for rice development; T_L is the lower temperature limit for rice development; T_O is the optimum temperature for rice development; D is the average day length in hours within the development phase; D' is

the critical day length in hours ($D' = 13h$); S_1 and S_2 are the beginning and end of the given phase; k, P, Q and G are initial values of the sub-model's parameters; and k', P', Q' and G' are, respectively, the final values of parameters k, P, Q and G.

$$\frac{dM}{dt} = \frac{1}{N} = \exp(k)\left(\frac{T - T_L}{T_O - T_L}\right)^P \left(\frac{T_U - T}{T_U - T_O}\right)^Q \exp[G(D - D')] \tag{1.20}$$

$$dM = \frac{dt}{N} = \exp(k)\left(\frac{T - T_L}{T_O - T_L}\right)^P \left(\frac{T_U - T}{T_U - T_O}\right)^Q \exp[G(D - D')]dt \tag{1.21}$$

$$\int_{S_1}^{S_2} dM = M = \sum_{i=1}^{N} \exp(k')\left(\frac{T - T_L}{T_O - T_L}\right)^{P'} \left(\frac{T_U - T}{T_U - T_O}\right)^{Q'} \exp[G'(D_l - D')] = 1 \tag{1.22}$$

Using the physiological leaf age sub-model, basically, the morphogenetic process can be determined by the number of leaves on the main culm [37] [183]. The basic mathematical model that uses j as the "leaf age" (in a physiological sense), when the jth leaf (L_j) is fully developed on the main culm; can be shown in equation (1.23). When $T = T_O$, and after arrangement (1.23) gives the equation (1.24). Where c, a, and b are initial values of the parameters.

$$L_j = \exp(-c)\left(\frac{T}{T_O}\right)^a N_j^b \begin{cases} T = 0 \ for \ T \leq T_L \\ T = T_O \ for \ T \geq T_O \end{cases} \tag{1.23}$$

$$N_{JO} = [\exp(C)L_j]^{1/b} \tag{1.24}$$

Indeed, the theoretical foundation of the rice clock model describes that the number of physiological days (DPD_l) required for the same developmental step is constant. DPD_l can be calculated by using equation (1.25); where T_l is mean daily temperature and D_l is daily day length. It is more reasonable than the concept used in the frequently applied degree-day method, where the summation of degree-days is assumed to be constant. In fact, the effect of day length on development is not considered and only a simple linear relationship between development rate and temperature is assumed in the degree-day method [54]. In addition, the actual total number of leaves in the main culm (L_t) can be calculated by equation (1.26), where L_w is whole leaf age and m indicates how many leaf ages the time interval would be equal to.

$$DPD_l = \left(\frac{T_l - T_L}{T_O - T_L}\right)^P \left(\frac{T_U - T_l}{T_U - T_O}\right)^{Q'} \exp[G'(D_l - D')] \tag{1.25}$$

$$L_t = L_w - m \tag{1.26}$$

The last one is the sub-model for organ development. There are six rules to define the sub-model, they are [54]:

1. Simultaneous with the emergence of the nth leaf, the (n–1)th leaf elongates
2. In parallel with the emergence of the nth leaf, the nth leaf sheath elongates
3. When the nth leaf emerges, a tiller starts emerging from the (n–3)th node
4. The number of nodes on the main culm corresponds to the number of leaves developed on the culm minus two
5. There exists a synchronous growth between the nth leaf and the (n–3)th roots
6. Synchronicity between leaf age (counted from the top of plant) and panicle development: 3.5th leaf (initial stage of panicle differentiation), 3rd leaf (primary branch differentiation stage), 2nd leaf (panicle primordia differentiation), 1st leaf (end of spikelet number increase), 0.5th leaf (initial stage of reduction division), 0th leaf (ripe pollen stage).

In 1997, [14] analyzed differences in competitiveness between two rice cultivars (IR8 and Mahsuri) and identified their major attributes with respect to competitive ability by using the INTERCOM model. Two rice cultivars, as object of the research, were taken from a field experiment that was conducted in the dry season of 1993 at the lowland research rice of the International Rice Research Institute (IRRI), Los Banos, Philipines. Indeed, there were four rice cultivars (IR8, IR50, Mahsuri and Oryzica Ilanos 5) observed in the field experiment, however only two rice cultivars with the lowest (IR8) and the highest (Mahsuri) competitive ability were used as object of this research; with concentrating in the routines for relating genotypic differences to phenological, physiological and morphological attributes.

The INTERCOM, as an ecophysiological process-oriented and compartment-based competition model, requires some inputs to operate, such as geographical latitude, daily weather data (minimum and maximum temperature, total global radiation), dates of crop and weed emergence, and crop and weed densities [89]. By constructing hypothetical IR8 isolines (replacing the parameter value of a single attribute by the value obtained

for Mahsuri), the model was used to determine how the differences in phenology, physiology and morphology lead to differences in competitive ability. The model also was able to show the strengths and limitations of ecophysiological competition models, with the background that it is very useful to design more competitive rice cultivars.

The analysis result showed that the competitive ability differences between IR8 and Mahsuri cultivars are mainly due to differences in maximum plant hight (H_{max}), the early leaf area growth rate (RGR_{la}), and relative early height growth rate (H_c). A prolonged vegetative growth gave an apparent increase in competitive ability, besides it increased the amount of vegetative biomass and consequently the respirative demand during grain filling. Interestingly, the model also showed that a higher light extinction coefficient resulted in poor light penetration and poor distribution of radiation within the canopy, thereby reducing radiation-use efficiency; on the other hand, thinner leaves resulted in earlier canopy closure and accordingly in an increase in grain yield. In addition, theoretically the competitive ability of rice has negative correlation with yield potential [76] [37]. The model confirmed this finding, that in weed-free conditions, the cultivar Mahsuri being the cultivar with the highest competitive ability, had lower yield than the other (IR8). From sensitivity analysis, simulated yield loss and weed shoot biomass were determined by increasing individual parameter values by 10%.

On one hand, the result shows that the model basically was able to quantify trade-offs between competitive and yielding ability of single traits; however, on the other hand, there are some situations where the model would not be able to quantify trade-offs. For examples the lodging of the tall-growing Mahsuri was not accounted for by the model; also the large fraction of unproductive tillers of Mahsuri was not considered in the model. This showed that a full quantitative estimation of trade-off was not feasible yet with the methods employed in the study [14]. However, FSPM (with 3-d representation of plant organs) technically can address the challenge.

Concentrating on canopy structure, [143] tried to design high-yielding rice plants. They traced the relationship between the solar energy (via photosynthesis) and the yield of rice plants in a quantitative way. Probably, this way can give a better understanding of core characteristics of rice plants capable of delivering substantially higher yields. In this study, by proposing a new advanced type of rice plant with very erect leaves and a large leaf area index (Vela), three laws of maximum yield were formulated; they refer to the maximum daily rice photosynthesis (P_gdmax), the maximum aboveground biomass (W_smax) and the yield of grain (W_gmax).

Regarding the relationship between canopy architecture and the distribution of solar irradiation, [143] concluded that the leaves of an erect canopy receive lower irradiance than the leaves of a prostrate canopy. The photosynthetically active radiation (PAR)

incident on a leaf (I_1 in μ mol $m^{-2}s^{-1}$) can be formulated with equation (1.27), where k is the extinction coefficient for PAR (m^2 ground m^{-2} leaf), I_0 is the irradiance incident on the canopy, L is the index of canopy, and τ is the fractional transmission of PAR through a leaf. Two types of canopy, canopies with erect leaves and canopies with more prostrate leaves, respectively have values of k between 0.2 – 0.3 and 0.6 – 0.8. In addition, the formula for describing the relationship between leaf photosynthesis (P_g) and irradiation (I_1) is described by equation (1.28); where α_T is the quantum yield at temperature T_{PS} that is calculated by using equation (1.29), and P_{max} is the theoretical rate of leaf photosynthesis.

$$I_1 = kI_0 \exp(-kL)/(1-\tau) \tag{1.27}$$

$$P_g = \alpha_\tau I_1 P_{max}/(\alpha_\tau I_1 + P_{max}) \tag{1.28}$$

$$\alpha_\tau = \alpha_{30} - 0.14(T_{PS} - 30) \tag{1.29}$$

Based on equation (1.29), the parameter T_{PS} (^oC) is needed. It is the mean daily temperature, a weighted mean of maximum and minimum temperatures (see equation (1.30)). If only mean temperature is available, then an alternative approach is needed to estimate the span between minimum and maximum temperature (T_{span}) by using the equation (1.31); where T_{span} is in the tropics as low as 5 ^oC, and T_{span} in subtropics could be 10 ^oC. To calculate the leaf photosynthesis (P_g) at irradiance (I) with unit g CH_2O m^{-2} ground, the equation (1.32) can be used. The maximum rate of canopy photosynthesis (P_{gdmax}) can be calculated by equation (1.33); where k_T is the quantum yield of the canopy ($g\ CH_2O\ MJ^{-1}$), and I_{day} is the total PAR incident on a rice plant in a day ($MJ\ m^{-2}d^{-1}$). The parameter k_T can be calculated by equation (1.34) where ϵ_c is a canopy efficiency factor (dimensionless).

$$T_{PS} = (3T_{max} + T_{min})/4 \tag{1.30}$$

$$T_{PS} = T_{mean} + 0.25T_{span} \tag{1.31}$$

$$P_g(I) = (P_{max}/k) \ln \{[\alpha_\tau kI + P_{max}(1-\tau)] \,/\, [\alpha_\tau kI \exp(-kL) + P_{max}(1-\tau)]\} \tag{1.32}$$

$$P_{gdmax} = 0.95 k_\tau I_{day} \tag{1.33}$$

$$k_T = \epsilon_c \alpha_\tau / (1 - \tau) \tag{1.34}$$

Furthermore, from the equation (1.35) can be seen that P_{gdmax} strongly affects the maximum aboveground biomass of rice (W_{smax}); where c_T is a temperature-dependent parameter. The parameter c_T itself can be calculated by equation (1.36), where f is a fraction of P_{gdmax} ($f \approx 0.7$) [142], β is a coefficient (0.59 in general), m_τ is a maintenance respiration term defined as the second phase of association with the maintenance of metabolic activity [139], and k_{dmax} is the fraction of dead matter in maximum aboveground dry matter (0.15 in general). The values of m_τ in tropics and subtropics respectively are 0.012 ($g\ g^{-1}\ d^{-1}$) and 0.0086 ($g\ g^{-1}\ d^{-1}$). Finally, the maximum grain yield (W_{gmax}) is linearly related to the maximum rate of canopy photosynthesis (see the equation (1.37)), where H is the harvest index [62]. The value of H is calculated by equation (1.38); where Y is grain dry weight, W_{sst} is the weight of the leaves and stems, and ρ is the weight of the panicle relative to grain weight.

$$W_{smax} = c_\tau P_{gdmax} \tag{1.35}$$

$$c_\tau = f\beta / (m_\tau (1 - k_{dmax})) \tag{1.36}$$

$$W_{gmax} = H c_\tau P_{gdmax} \tag{1.37}$$

$$H = Y / (W_{sst} + \rho Y) \tag{1.38}$$

Two important conclusions, especially in morphological aspects, of this study could be taken. The first, the improvement of rice plant yield can be made by improving the canopy photosynthesis, where the improvement has to be focused on canopy architecture and yield of photosynthesis. The second is that improvement in canopy architecture should be focused on Vela plants (as new advanced type of rice plant) with sufficient vegetative and productive sink capacity to realize the benefits of improved carbon capture [143].

In a meta-analysis, [140] have reviewed about 140 scientific articles about rice (*Oryza sativa* L.) and maize (*Zea mays* L.). Especially for articles about rice, those articles were mostly extracted from scientific databases, such as Google Scholar, CAB Abstract, Web of Science, Agris and Agricola. The published material was made available by the Faculty of Life Science library at the University of Copenhagen. In the review, [140] classified temperature data into three categories: optimum temperature, giving the highest rate of a crop process; minimum temperature, as a lower limit; and maximum temperature, as an upper limit. Temperature was a focused parameter that they used in the review, because of one reason, namely that the temperature strongly affects plant growth and development [152] [11]. In addition, they described the responses of plants to extreme temperatures under experimental condition.

Particularly for rice plants in the vegetative stage, based on two sub-species (japonica and indica), several conclusions can be taken. Mean lethal temperatures are 4.7^oC (SE 1.2^oC) for minimum temperature and 42.9^oC (SE 0.7^oC) for maximum temperature. The mean temperatures for the leaf initiation process are 10^oC (SE 0.6^oC) and 42.5^oC (SE 2.5^oC) for minimum and maximum temperature respectively. For the shoot growth process, the needed mean temperatures are between 13.7^oC (SE 2.1^oC) and 35.5^oC (SE 0.5^oC). Especially for the root growth process, the required mean temperatures are 15.8^oC (SE 0.8^oC) and 35.9^oC (SE 0.6^oC) for minimum and maximum temperature respectively. Lastly, the range of mean temperature for the development of the whole plant is between 13.5^oC (SE 2.1^oC) and 35.4^oC (SE 2.0^oC), with the optimum at 27.6^oC (SE 2.0^oC); where SE of maximum temperature is small, mostly because the rice is susceptible to heat damage during the vegetative stage when temperature is between 40^oC and 45^oC.

1.4.2.2 Morphology and 3d Models of Rice Plant Development

[75] analyzed the morphology and development dynamics of tillers of an Indian cultivar of rice plant (Ir64). Here, the development dynamics was investigated and the structural and temporal characteristics of tillers were examined. Both investigation and examination were based on topological location of tillers, the timing of appearances and main stem (MS) development stage. The study was conducted in a glasshouse at the Institut de Recherche pour le Dévelopement (IRD) in Montpellier (France). The temperature was 25^oC in the day time and 20^oC at night.

The analysis yielded the following results. The rice population was clustered into three categories of phytomer numbers on the MS; 15, 16 and 17 phytomers on the MS. The number of phytomers on the different axes decreases with branching order (order 1 >

order 2 > order 3 > ...) and with increasing tiller rank (T3 > T4 > T5 > ...). Regarding the tiller number and position, the MS supports 5.4 ± 0.81 primary tillers and 6.1 ± 1.87 secondary tillers; or on average, one rice plant has 11.4 ± 3.01 tillers (without tertiary tillers). Tillers develop between ranks 3 and 10 (where tillers before rank 3 and after rank 10 are dormant), and mostly flower before rank 9. On the other hand, secondary tillers appear from phytomers 2 (Tx2) to 5 (Tx5) of the primary tillers, it happens for 98% of tillers of rank 4 (T4) to 6 (T6); only very few secondary tillers develop on rank 3 and rank 7 of primary tillers.

The used numbering and notation for branching orders and ranks of leaves and tillers can be seen in Figure 1.13. Tiller or leaf position is defined by the rank of the axillating phytomer (F) on the bearing axis. A primary tiller is defined by the 'x' phytomer on the MS designed 'Tx' and a secondary one is defined by the 'y' phytomer on the Tx tiller designed 'Txy'. The MS, primary tillers and secondary tillers are assigned to order 1, order 2 and order 3 respectively. Signs '<' and '+' denote succession and branching, respectively.

In addition, the first primary tillers emerge 24 days after sowing, and continue to emerge until the 65th day (for T10). Tillers T4 and T5 appear at the same time as leaf 7 and leaf 8 on the MS; and in a few cases, T3 appears at the same time as T4. Moreover, the secondary tillers emerge between 34 and 76 days after sowing, and they are visible at a precise stage of MS development (between emergence of leaf 10 and leaf 11). In addition, the emergence of the first tillers occurred at the 7-leaf emergence on the MS. Besides that, regarding the leaf; the visible tip of a new leaf blade appears when the preceding sheath leaf has completely expanded. There are six leaves on the MS after 20 days and nine leaves after 34 days. The first 12 leaves completely expand on the MS after 48 days, and the 15th leaf appears 35 days later. The leaves increase linearly during 45 – 50 days after sowing. The leaf blade length on the MS increases regularly up to leaf ranks 12 and 13, and then decreases rapidly until the last leaf.

Regarding flowering of tillers, inflorescences (I, see Figure 1.14) emerge between 97 and 111 days after sowing, and 71% of primary tillers flower. The synchronous appearance of tillers and leaves on the MS throughout plant development until flowering, and a relation between emergence time and tiller flowering were noticed by [75] as important characteristics of rice development. Also, the decrease in leaf size and the death of late tillers occurred at the floral initiation stage, identified when 12 – 13 leaves are visible on the MS. This points to a tradeoff in assimilate supply between vegetative and reproductive organs.

By using the L-system formalism, [175] developed 3-d virtual rice plants to specify rice plant architecture and to find appropriate functions in presenting all growth stages of

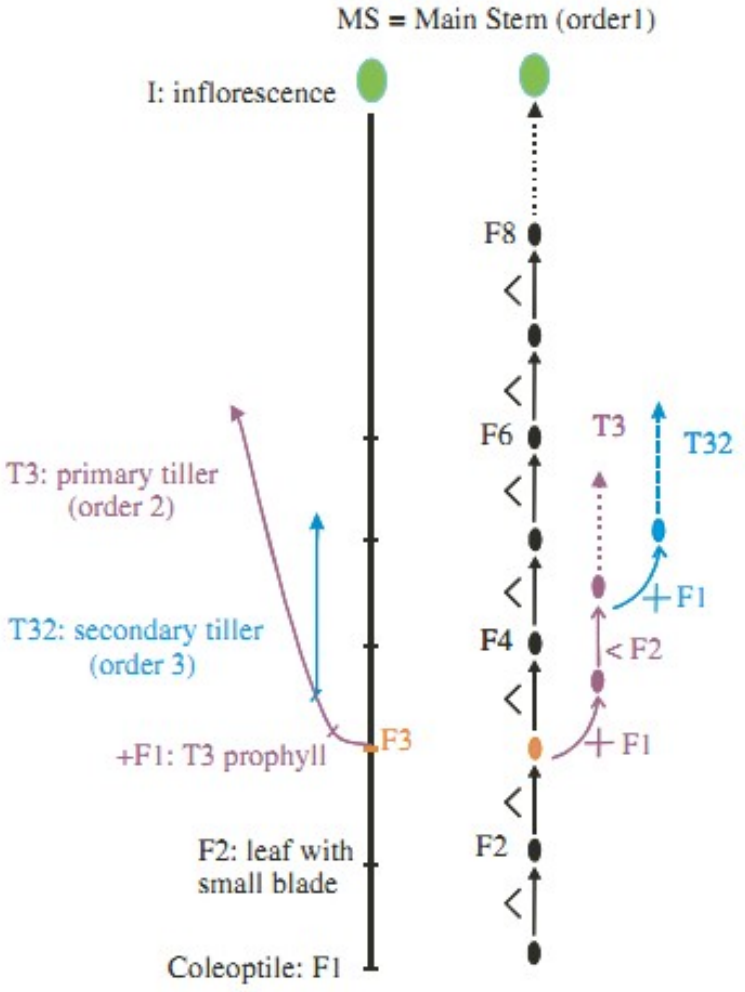

FIGURE 1.14: Diagrammatic of representation of rice plant [75]

rice plants. Japonica-type rice plants were used as object in the research, they were grown in the open at CSIRO Long Pocket Laboratories, Brisbane, Australia.

Some parts of the model were described in detail in the study, such as development stage, leaf number, vegetative development, leaf blade length, leaf blade width, leaf angles, leaf azimuth, leaf sheath length, internode length, relationship between the main stem and a tiller, and tiller angles. In the development stage part, [175] used the development index (D_V) from [117]. D_V is an accumulation of development rate (D_R), which is a function of daily mean temperature (T) and photoperiod (L). It can be calculated by equations (1.39) up to (1.43); where G_j is the minimum number of days required for completing each phase (for vegetative phase $j = 1$, for reproductive phase $j = 2$); D_{V*j} marks the start ($j = 1$) and end ($j = 2$) of the photosensitive phase; a_j, T_{hj}, T_c, b_j and L_c are parameters for determining the shape of the functions.

when $D_V < 1.0$

$$D_R = \begin{cases} f_1(T), \text{ when } D_V < D_{V*1} \\ f_1(T)g_1(L), \text{ when } D_V \geq D_{V*1} \end{cases} \tag{1.39}$$

when $1.0 \leq D_V < 2.0$

$$D_R = \begin{cases} f_2(T), \text{ when } D_V < D_{V*2} \\ f_2(T)g_2(L), \text{ when } D_V \geq D_{V*2} \end{cases} \tag{1.40}$$

when $2.0 \leq D_V \leq 3.0$

$$D_R = a_3(T - T_c) \tag{1.41}$$

$$f_j(T) = (1/G_j)/1 + \exp[-a_j(T - T_{hj})], \text{ when } D_V \leq 2.0 \tag{1.42}$$

$$g_j(L) = \begin{cases} 1 - \exp[b_j(L - L_c)], \text{ when } L < L_c \\ 0, \text{ when } L \geq L_c \end{cases} \tag{1.43}$$

Furthermore, leaf number (L_N) can be calculated by equation (1.44) for $0 < D_V < 1$ (where the leaf emergence rate L_R is given by 1.45) and equation (1.46) for $1 \leq D_V < 2$; where G_L, A_L, T_L, C_L, D_L, and E_L are parameters; T is the daily mean temperature; L_{N0} is the initial leaf number at transplanting (in this case 1). Then the vegetative development was specified in the form of L-system rules. Translated to XL syntax, vegetative development is formulated by rule (1) for order = 0 (i.e., for the main stem) and rule (2) for order = 1; where $ApicalMeristem$ is the apical meristem module; $Internode$ is the internode module; $AxillaryBud$ is the axillary bud module; $LeafSheath$ is the leaf sheath module; $LeafBlade$ is the leaf blade module; i, j, k, and l are parameters indicating the node of attachment to the main stem, primary tiller, secondary tiller and tertiary tiller respectively; and $tsum$ is the cumulative degree-days above a base temperature from start of leaf emergence.

$$L_N = \sum L_R + L_{N0} \tag{1.44}$$

$$L_R = (1/G_L)\left\{1 + \exp[-A_L(T - T_L)]\right\}\left\langle (1 - C_L)/\left\{1 + \exp[B_L(D_V - D_{VL})]\right\} + C_L\right\rangle \tag{1.45}$$

$$L_N = leafnumber(atD_V = 1.0) + 3.5 - \min\left\{0, D_L[1 - \exp\left\{-E_L(D_V - 1)\right\}] - 3.5\right\} \tag{1.46}$$

Rule (1)

```
ApicalMeristem(order, i, j, k, l, tsum, DV), (order == 0) ==>
        Internode(order, i, j, k, l, 0, DV)
        [AxillaryBud (order+1, i+1, j+1, k, l, 0, DV)]
        [LeafSheath (order, i+1, j, k, l, 0, DV)
        LeafBlade(order, i+1, j, k, l, 0, DV)]
        ApicalMeristem (order, i+1, j, k, l, 0, DV)
```

Rule (2)

```
ApicalMeristem(order, i, j, k, l, tsum, DV), (order == 1) ==>
        Internode(order, i, j, k, l, 0, DV)
        [AxillaryBud(order+1, i, j+1, k+1, l, 0, DV)]
        [LeafSheath(order, i, j+1, k, l, 0, DV)
        LeafBlade(order, i, j+1, k, l, 0, DV)]
        ApicalMeristem (order, i, j+1, k, l, 0, DV)
```

The leaf length at position n (L_n) can be calculated by equation (1.47) with L_{max}, as the maximum length of any blade in the same axis, to be calculated by equation (1.48), and d, as a parameter that adjusts the width of the bell-shaped function, can be calculated by equation (1.49); where L_{max} and d are functions of the attachment position of the tiller to the main stem (P_t). Then, the leaf blade width can be calculated by equation (1.50); where W_p is the limit of the maximum width; and b, c and d are parameters. And, for the fully expanded internode length ($I_{max,nf}$), it can be calculated by equation (1.51), with the proportionality factor (r_{nf}) being calculated by equation (1.52); where I_{total} is the sum of all internode lengths, and nf is the internode number (it is counted from 1 as the first internode to elongate). However, this relationship was only established for the final four internodes on the main stem.

Moreover, the leaf angles (the axial angles between sheaths and blades of leaves) vary with position between 40^o and 15^o (on main stem), between 75^o and 12^o (in primary tillers), and between 60^o and 15^o (in secondary tillers). The leaf sheath length is calculated by using equation (1.53) with n representing positions starting from 1 at the second leaf on the main stem (because the first leaf on the main stem has no blade [10]. Furthermore, for leaf azimuth, each leaf azimuth has some random deviation from the basic alternating phylloxis, with RH(normal(0,15)) representing a random deviation of each leaf azimuth with mean = 0 and standard deviation = 15^o.

$$L_n = L_{max} \exp\left[\left(-(n - nmax)^2\right)/(2d)\right] \tag{1.47}$$

$$L_{max} = 489 - 29.1P_t \tag{1.48}$$

$$d = 3.44 - 0.23P_t \tag{1.49}$$

$$W_{max} = W_p/[1 + \exp(b - cL_{max})]^{1/d} \tag{1.50}$$

$$I_{max,nf} = r_{nf} \times I_{total} \tag{1.51}$$

$$r_{nf} = 0.051nf^{1.62} \tag{1.52}$$

$$S_{max,n} = 41.4 + 0.393L_{max,n} \tag{1.53}$$

For the relationship between the main stem and a tiller, the variations of axial angles used are between 2^o until 32^o approximately. The schematic figure of the relationship can be shown in Figure 1.15; where the angle between line segments 0 and 1 is the maximum angle between main stem and the primary tiller; and the angle between line segments 0 and 2 is the tiller angle after internode elongation. And, the axial angle between a parent module and its daughter tiller (θ_{till}) increases with the number of tillers produced by the parent axis above and on the same side as the daughter tillers. The angle is calculated by equation (1.54), where N_p is the number of those daughter tillers, N_d is the number of tillers produced by the daughter tiller, and a_{till} is a constant.

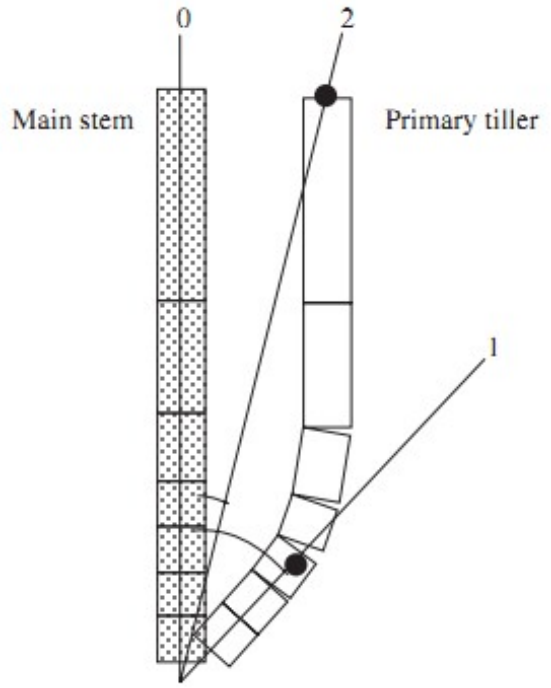

FIGURE 1.15: Schematic view of relationship between main stem and a tiller [175]

$$\theta_{till} = a_{till}(N_p + N_d) \tag{1.54}$$

The simulated number of tillers showed a constant value after reaching the maximum tillering stage, whereas the observed number of tillers decreased until flowering. The model underestimated the leaf number, thus the actual development of leaves was slightly faster than the model output; and also, the simulated leaf number was usually smaller than the measured [175].

Concentrating on the shape of leaves, [187] proposed a new quantitative approach for evaluating the degree of bending of grass-type leaves, especially in leaves of rice. The new quantitative approach is based on P-type Fourier Descriptors (PFDs). They implemented the approach in eight rice cultivars: Jakov, Karahoushi, Koshihikari and Nipponbare (four Japonica cultivars); IR64, MTL250 and THMT (three Indica cultivars); and one Glaberrima cultivar.

The PFDs is a method to represent, retrieve and normalize shape [185]. In this case, [187] fitted the PFDs to the shape of rice plants leaves. The shape is represented by a sequence of coefficients. The PFDs are very effective to be fitted to open curve shapes. The curve can be figured by many straight lines in a digital image as shown in Figure 1.16. $a(j)$ is the angle of deviation at point $z(j)$; where $a(0)$ is determined as the angle

between the vector $z(1) - z(0)$ (Figure 1.16(a)), and the x-axis ($a(j)$) is defined as the angle between the vector $z(j+1) - z(j)$ and the vector $z(j) - z(j\text{-}1)$ (Figure 1.16(b)), and accumulated angles are defined in equation (1.54); where θ(j) is the angle between the x-axis and $z(j+1)$-$z(j)$ (Figure 1.16(c)). In addition, the inverse Fourier transform ($w(j)$) is defined by equation (1.55); where δ is the length of the vector defined as $\delta = \|z(j+1) - z(j)\|$ for $j = 0, 1, \ldots, n-1$. Furthermore, the discrete Fourier transform of w(j) is defined in equation (1.56); where n is the number of points of measurement. The curves $C_p(k)$ are usually called P-type Fourier descriptors. Then, from the equation (1.57), the inverse Fourier transform can be derived as equation (1.58).

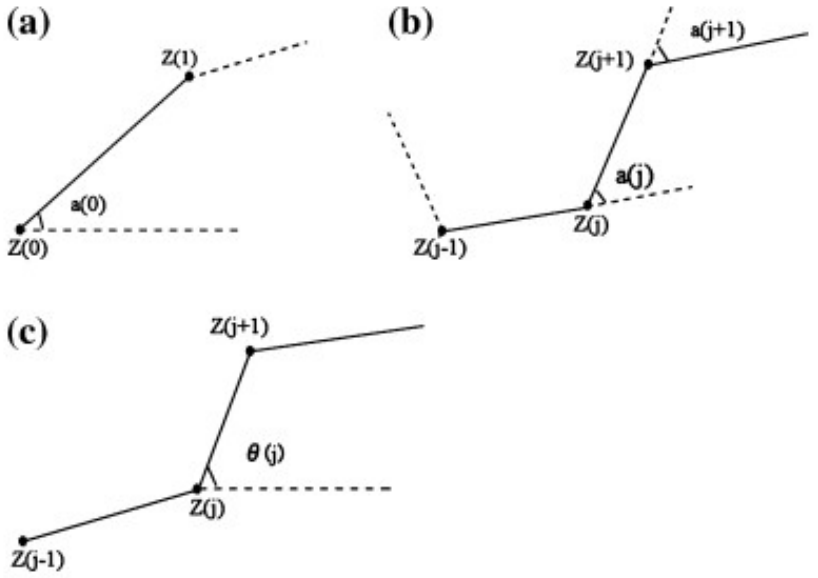

FIGURE 1.16: Angle definition for P-type Fourier Descriptors [187]

$$\begin{cases} \theta(0) = a(0) \\ \theta(j) = \theta(j-1) + a(j), \ j = 1, ..., n-1 \end{cases} \tag{1.55}$$

$$w(j) = \exp(i\theta(j)) = \cos\theta(j) + i\sin\theta(j) = \frac{x(j+1) - x(j)}{\delta} + i\frac{y(j+1) - y(j)}{\delta} \tag{1.56}$$

$$C_p(k) = \frac{1}{n}\sum_{j=0}^{n-1} w(j) e^{-i\frac{2\pi}{n}kj}, \ (k = 0, 1, ..., n-1) \tag{1.57}$$

$$w(j) = \sum_{k=0}^{n-1} C(k) \exp\left(2\pi i \frac{jk}{n}\right) \tag{1.58}$$

The result can be seen in Figure 1.17, showing the redrawn contour of the object. To redraw the curves in terms of PFDs, σ' is used to describe the average leaf length. σ' is formulated by equation (1.59); where i refers to the ith cultivar (i=1,2,...,8). The PC1 to PC4 indicate the principal components. The broken line and dotted line show the cases where the PFDs are modified by +2 SD and -2 SD (standard deviation) respectively. The first row and second row illustrate the first and the second leaf.

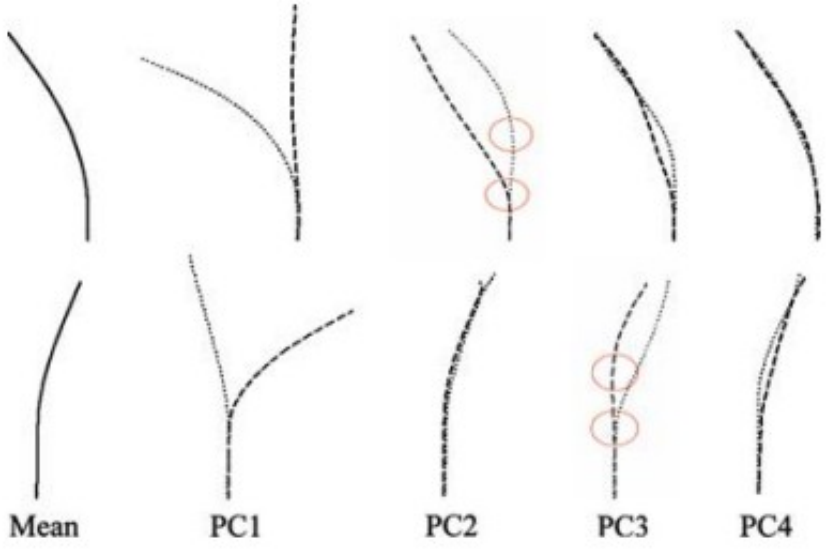

FIGURE 1.17: Image description for main principal components [187]

$$\sigma' = \frac{average\ length\ of\ i\ cultivar's\ leaves}{average\ length\ of\ all\ leaves} \times \sigma \tag{1.59}$$

Some points were concluded. The curves that result from redrawing utilizing only the lower frequency part of the PCs are extremely stable. The tip sections of several of the second leaves have withered, which was reflected in one of the principal components of the PFD representation. Two of the principal components represented the bending rate and the connector point of leaf to tiller, respectively.

Also [189] developed a leaf shape model of the rice plant (*Oryza sativa* L.) to characterize the change pattern of leaf growth during plant development and to model the changes in morphology of the different leaves as well. Basically, leaves are the main photosynthetic organs that have most impact on light energy utilization. This is one reason why the model is very fruitful for designing optimal plant shape and visualizing plant growth.

The model was developed based on four experiments carried out in 2005 at the field station of Nanjing Agricultural University, China. In the first experiment, the effect of four nitrogen levels was investigated (0, 82, 165 and 247 $kg\ N\ ha^{-1}$), by using urea

as N-fertilizer. In the second experiment, four water regimes were established (70%, 80%, 90% and 100% of soil capacity). These water regimes were applied in four kinds of cultivar of rice (the third experiment): Wuxiangjing 14, Nippobare, Nanjing 16 and Yangdao 6. Finally, treatments consisting of factorial combinations of nitrogen rates and water regimes were applied in the fourth experiment. This study of [189] was adopted as main method of our model, thus the detail explanation about several equations are delivered in chapter 6.

[24] tried to identify the relationships between rice plant architecture parameters and corresponding organ biomass by developing a model of biomass-based rice architectural parameters. The model also can be used in morphological models. Basically, there are three phases, stages of growth in whole rice life, each with its own morphogenesis rules: seedling, from emergence to elongation; mid, from elongation to heading; and late stage, from heading to maturity.

Here, [24] described the morphogenesis of the young seedling stage (from emergence to early tiller). They developed five types of model: leaf blade length, the maximum width, sheath length of fully grown leaves, bowstring length, and leaf angle models. In their leaf blade length model, the jth leaf blade length ($LL_j(i)$) on the main stem on the ith day after emergence (cm) is described in equation (1.60); where, $DWLB_j(i)$ is the jth leaf blade dry weight (g), it is formulated in equation (1.61); $RLW_j(i)$ is the ratio of the jth leaf blade length to blade dry weight ($cm\ g^{-1}$), it is close to a fitted cubic polynomial curve function; $CPLB_j(i)$ is the ratio of the jth leaf blade dry weight to that of the whole single aboveground plant ($g\ g^{-1}$) on the ith day after emergence, it is close to a sigmoid shape function; $DW_{sp}(i)$ is the dry weight per plant on the ith day after emergence (g / plant), it is formulated in equation (1.62); $MDW_{sp}(i)$ is the mean dry weight per plant on the ith day after emergence (g / plant), it is described in equation (1.63); $SDW_{sp}(i)$ is the standard error of dry weight per plant on the ith day after emergence (g / plant); $DW_{cp}(i)$ is the dry weight in canopy per area on the ith day after emergence ($g\ m^{-2}$); and DES is the plant number per area (plant m^{-2}). $RLWj(i)$ itself is described in equation (1.64); where $Lpji$ is the jth leaf position on the main stem on the ith day after emergence; and B_0, B_1, B_2, and B_3 are unstandardized coefficients. In addition, $CPLB_j(i)$ is given in equation (1.65); where b_0 and b_1 are coefficients.

$$LL_j(i) = DWLB_j(i) \times RLW_j(i) \tag{1.60}$$

$$DWLB_j(i) = CPLB_j(i) \times DW_{sp}(i) \tag{1.61}$$

$$DW_{sp}(i) = MDW_{sp}(i) \pm SDW_{sp}(i) \tag{1.62}$$

$$MDW_{sp}(i) = DW_{cp}(i)/DES \tag{1.63}$$

$$RLW_j(i) = B_0 + B_1 LP_{ji} + B_2 LP_{ji}^2 + B_3 LP_{ji}^3,\ 1 \leq j \leq 6 \tag{1.64}$$

$$CPLB_j(i) = e^{b_0 + b_1/LP_{ji}},\ 1 \leq j \leq 6 \tag{1.65}$$

Regarding the maximum leaf blade width model, the jth maximum leaf blade width ($LW_j(i)$) on the ith day after emergence (cm) is given in equation (1.66). Furthermore, the jth leaf sheath length ($LS_j(i)$) of fully grown leaves on the ith day after emergence (cm), in the leaf sheath length model, can be described in equation (1.67); and the jth leaf blade bowstring length ($LBBL_j(i)$) on the ith day after emergence (cm), in the leaf blade bowstring length model, is given in equation (1.68). And then, in the leaf blade angles models, the blade tangent angle (TA_j, in ^{O}) and blade bowstring angle (BA_j, in ^{O}) are given in equations (1.69) and (1.70); where $RTW_j(i)$ and $RBW_j(i)$ are the ratio of the blade tangent angle and the blade bowstring angle to the jth leaf blade dry weight on the main stem on the ith day after emergence ($^O g^{-1}$). The $RTW_j(i)$ and $RBW_j(i)$ are described in equations (1.71) and (1.72).

$$LW_j(i) = e^{b_0 + b_1/LL_{ji}},\ 1 \leq j \leq 6 \tag{1.66}$$

$$LS_j(i) = b_0 LL_j(i)^{b_i},\ 1 \leq j \leq 5 \tag{1.67}$$

$$LBBL_j(i) = b_0 + b_1 LL_j(i),\ 1 \leq j \leq 5 \tag{1.68}$$

$$TA_j(i) = DWLB_j(i) \times RTW_j(i) \tag{1.69}$$

$$BA_j(i) = DWLB_j(i) \times RBW_j(i) \tag{1.70}$$

$$RTW_j(i) = b_0 LP_{ji}^{b_1},\ 1 \leq j \leq 5 \tag{1.71}$$

$$RBW_j(i) = b_{B0} LP_{ji}^{b_{B1}},\ 1 \leq j \leq 5 \tag{1.72}$$

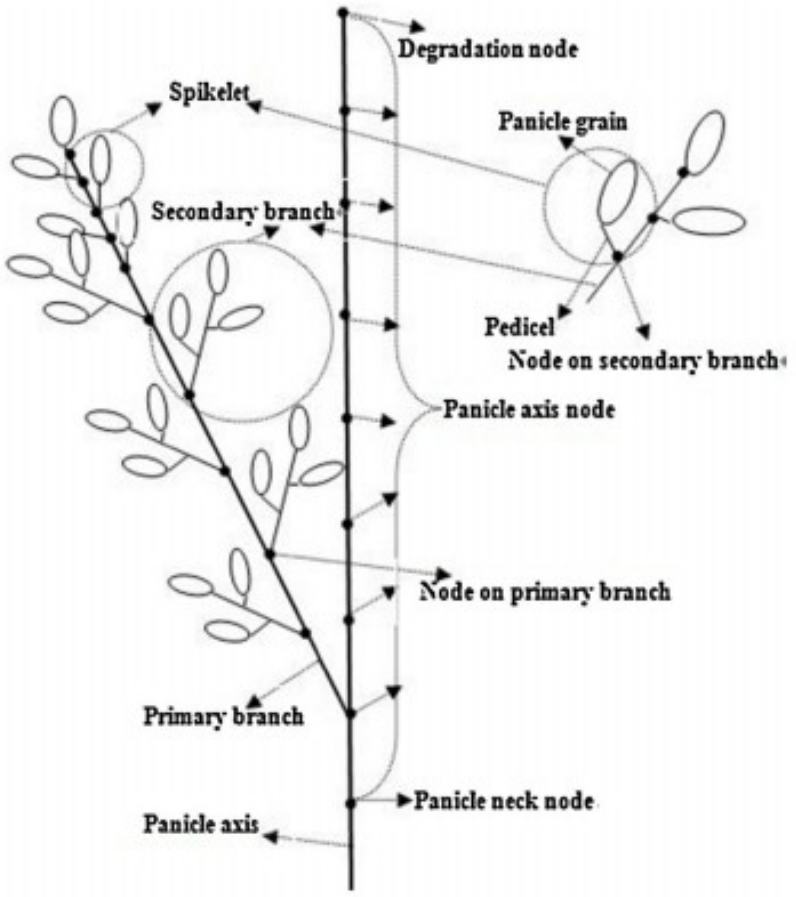

FIGURE 1.18: Topology of rice panicle [186]

Finally, we shortly review [186]. They developed a model for predicting the dynamics of panicle geometric morphology of rice in three dimensions. Theoretically, the topological structure of a rice panicle can be shown in Figure 1.18. The ratio of the number of secondary branches on the primary branch n ($NSBPB_n$) to the total number of primary branches ($TNPB$) changes with the panicle axis node and can be described by equation (1.73); where $NSBPB_a$ is an equation coefficient, $NSBPB_b$ is the ratio of the secondary branch number on the primary branch n_1 to $TNPB$, n_i is the rank of the axis node with the highest secondary branch number. The number of spikelets per unit length of secondary branch ($NSULSB$) can be calculated by equation (1.74); where $TNPB$ x $ANSPB$ is the number of spikelets on all primary branches, $ASBL_n$ is the average

length of secondary branches on primary branch n. $ASBL_n$ itself can be defined by equation (1.75); where PBL_n is the length of primary branch n, $ASBL_a$ is an equation coefficient, n_3 is the rank of the axis node where the average length of secondary branches is minimum, and $ASBL_b$ is the minimum value of the ratio ($ASBL_n$ / PBL_n).

$$\frac{NSBPB_n}{TNPB} = NSBPB_a \times (n - n_1)^2 + NSBPB_b \tag{1.73}$$

$$NSULSB = \frac{TNS - TNPB \times ANSPB}{\sum_{n=1}^{TNPB} NSBPB_n \times ASBL_n} \tag{1.74}$$

$$\frac{ASBL_n}{PBL_n} = ASBL_a \times (n - n_3)^2 + ASBL_b \tag{1.75}$$

1.5 Optimization Methods

Basically, optimization is an act to obtain the best result based on given circumstances [133]. In more technical terms, the optimization is one technique or method to find the values of a set of parameters which maximize or minimize the objective function of interest [43]. The purpose of the optimization itself is to select the minimum or maximum one of some possible resulting values of a mathematically defined objective function. The term circumstance in optimization, is the crucial term; as it can represent and also restrict the objective function. Practically, the circumstance can be symbolized by selected parameters and their allowed ranges of values; where the selected parameters and their restrictions have strong effect to define the specific objective function.

Connected with the optimization issue, numerous statistical approaches theoretically can be used to find a set of parameters in obtaining the best (optimal) value of the objective function. One of them is a full factorial design (it is also called "a level to the factor design" or $(level)^{factor}$ full factor design). It is an approach or method that can be used in optimizing by exploring the whole parameter space. All possible parameter combinations in parameter space are scanned and tested to see the resulting value of the objective function. The number of parameters and their granularity (size of interval between successive values) strongly affect the time consumed in the optimization process.

Sampling is a method that commonly can be used to avoid a full factorial design in scanning all possible parameters in optimizing. Two kinds of sampling method that will be explained here are a Simple Random Sampling (SRS) and Latin Hypercube Sampling (LHS). In SRS, each element (parameter combination) has an equal probability of being

selected from all population units; it is also called equal probability of selection method [4]. There are two types of SRS: SRS with and without replacement. In SRS with replacement, the sampled unit (value) is randomly selected from the population and it is replaced again in the population before next selection. In contrast, the SRS without replacement does not replace the sample unit before next selection. However, generally the SRS method is not efficient for optimizing; as sometimes it looks similar to the full factorial design, where the sampling points are spread over the parameter space but leave broad gaps [157].

Concerning LHS, it is a sampling method to scan parameter space which tries to eliminate biased choices of parameter values. Technically, its purpose is to ensure that each value (or a range of values) of a parameter is represented in samples ([109] in [28]). For instance, to generate n samples from 2 parameters (X, Y $\in$ [0, 1]); firstly we have to divide the ranges of both X and Y into n equal intervals (n^2 cells). This is shown in Figure 1.19 [28].

n					
...					
3					
2					
1					
X/Y	1	2	3	...	n

FIGURE 1.19: A Latin Hypercube Sample scheme on the unit square, with n levels for each parameter [28]

The basic requirement of the LHS method is that each row and column of the constructed table (Figure 1.19) contains only one sample; for instance, the Figure 1.20 requires n samples. For each sample ([i, j]), the sample values of X and Y can be identified by equations (1.76) and (1.77); where ξ_X and ξ_Y are random numbers (ξ_X, ξ_Y ϵ [0, 1]), and F_X and F_Y are the cumulative probability distributions functions of X and Y respectively. Thus, the final result has to be performed in one matrix called a Latin Hypercube matrix ($LHS_{m,n}$); where m indicates a parameter and n is the net work sample size. The matrix is shown in Figure 1.20 [28].

$$X = F_X^{-1}\left((i - 1 + \xi_X)/n\right) \tag{1.76}$$

$$Y = F_Y^{-1}\left((j - 1 + \xi_Y)/n\right) \tag{1.77}$$

Sample	X	Y
1		
2		
3		
...		
n		

FIGURE 1.20: A Latin Hypercube Matrix ($LHS_{m,n}$) [28]

In addition, actually there is no single optimization method or technique that can solve all type of optimization problems [133]. The optimization method specifically is constructed only for the specific problem in a specific case. The capability of developing the appropriate method is exceptionally mandatory here.

Three kinds of methods that can be used to find a local optimum (a local maximum or minimum) of a function are Gradient, Quasi-Newton, and hill-climbing methods. They are used to find stationary points of a function, i.e., points where the gradient of a function is zero. The gradient of functions ($f(x)$) is denoted as $\nabla f(x)$; and it is defined in equation (1.78) [20]. Where $\frac{d}{dx}$ denotes the derivation operator and n is the dimension of the parameter space.

$$\nabla f(x) = \left(\frac{df(x)}{dx_1}, ..., \frac{df(x)}{dx_n} \right) \tag{1.78}$$

Theoretically, the gradient method is an algorithm to find extremal points of a function; with the search directions defined by the gradient of the function at the current point [126]. Two types of gradient method are a gradient descent (with step size γ proportional to the gradient) and a conjugate gradient (by combining the information from all previous search directions) [161]. In addition, the Quasi-Newton method is another optimization method that is iterative (involving a series of line searches) and generally involves computation only of $f(x_k)$ and $\nabla f(x_k)$ at each iteration. It is much like Newton's method; where the basic form of the algorithm is shown below [21]:

1. Make an initial guess x_0 at the minimum; set $k = 0$. Initiate $H_0 = I$ (the n x n identity matrix)
2. Compute $\nabla f(x_k)$ and take the search direction as $h_k = -H_k \nabla f(x_k)$
3. Do a line search from x_k in the direction h_k and take $x_{k+1} = x_k + t * h_k$, where $t * h_k$ minimizes $f(x_k + th_k)$

4. Compute H_{k+1} by modifying H_k appropriately. This is usually done by setting $H_{k+1} = H_k + U_k$, where U_k is some easy to compute 'updating' matrix. Set $k = k + 1$ and go to step 2

Where h_k is a search direction, and H_k is a Hessian matrix. On the other hand, geometrically $(x_{n+1}, 0)$ is derived from the line tangent of the function $f(x)$ at $(x_n, f(x_n))$. See Figure 1.21, it is an interaction between line tangent and function in a special case (dimension 1).

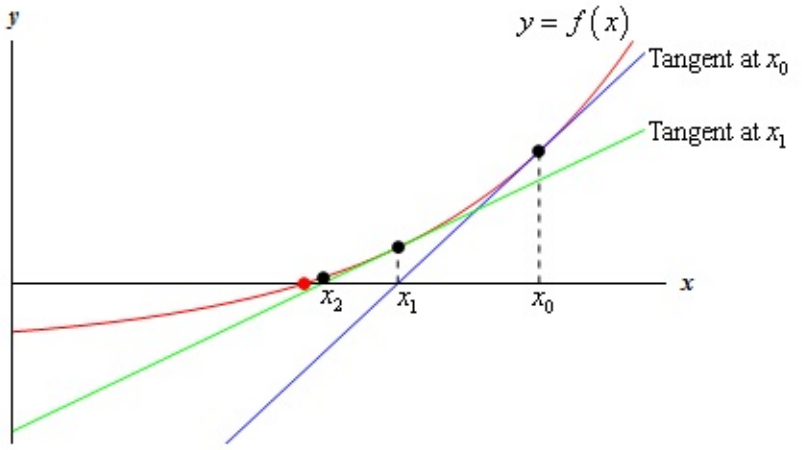

FIGURE 1.21: Interaction between line tangent and function $f(x)$ [36]

Especially for the hill-climbing method, by using a simple loop that is continuously able to move to the next increasing or decreasing value of the goal function (neighbor), it can reach a local optimum (maximum or minimum). This is a simple method that has been successfully applied to solve many problems of optimization [137]. Practically, the hill-climbing method can be applied in both optimization problems of discrete and continuous parameters.

One optimization method that can be realized to avoid getting stuck in a local optimum which is not the global optimum is the Simulated Annealing method. It is a stochastic method used to find the global optimum. The basic idea of the simulated annealing method to escape from local optima is realized in the execution process. The simulated annealing is executed until an external "termination condition" is satisfied; where the parameter temperature (T) is diminished periodically [114]. In the optimization algorithm, the parameter T is used to control the searching range, this means if parameter T has the biggest value (in this case, one is the biggest value); all points in parameter space have to be visited as optimum point candidates. The algorithm pseudocode of the simulated annealing is presented in Listing 1.1.

```
Procedure SimulatedAnnealing
Begin
    X = some initial starting points in S
    While not termination-condition Do
        X = improve (X, T)
        Update (T)
    End While
    Return(X)
End
```

LISTING 1.1: The pseudocode of simulated annealing procedure

```
Procedure Genetic Algorithm
Begin
    Generate random population of chromosomes
    Do
        Evaluate the fitness of each chromosome in population
        If (fitness value fulfills selection criterion)
            Crossover
            Mutation
            Place new offspring in a new population
        End if
        Use new generated population
    Until condition is satisfied
End
```

LISTING 1.2: The pseudocode of basic genetic algorithm

In addition, another class of optimization methods are Evolutionary Algorithms (EAs) or Genetic Algorithms (GAs). Conceptually, GAs are a particular subclass of EAs. They use techniques inspired by evolutionary biology, such as inheritance, mutation, selection and crossover (also called recombination) [169]. EAs are population-based meta heuristic optimization algorithms that use biology-inspired mechanisms and survival of the fittest theory in order to refine a set of solutions iteratively [106].

In other words, GAs theoretically can be defined as computer based search techniques patterned from the genetic mechanisms of biological organisms that have adapted and flourished in a changing highly competitive environment [106]. GAs can be used to solve hard optimization problems quickly, often reliably and accurately by using genetic process concepts (such as selection or mutation) [157] [106]. However, like hill-climbing and simulated annealing, GAs belong to the class of heuristic methods; there is no guarantee that they will always deliver the globally-best solution of an optimization problem. The basic GA can be structurally defined by Listing 1.2.

Technically, the population is a set of individuals each representing a possible solution to a given problem. The chromosome consists of genes joined together to form a string of values, where a gene is a parameter that contributes to describe a solution to the problem. The chromosome is sometimes called a genome and in programming also defined as a string of binary symbols or other data structures; and the gene itself is defined as a character or number. In addition, the fitness score (value) is a score that is inferred from the chromosome itself by using a fitness function, it is sometimes indicated by a

real number (value between 0 and 1; and 1 indicates the highest fitness value) which reflects the utility or the ability of the individual which that chromosome represents; where the fitness function evaluates each solution to decide whether it will contribute to the next generation of solutions. The fitness function tightly depends on the application case, it can calculate: strength, weight, width, maximum load, cost or combination of all these. Furthermore, the generation is indicated by the execution of the loop of the above algorithm when it was created. The crossover decomposes two distinct solutions and then randomly mixes their parts to form novel solutions; it means choosing a random position in the string and exchanging the segments either to the right or to the left of this point with another string partitioned similarly to produce two new offsprings. On the other hand, mutation randomly perturbs a candidate solution in the case of binary genes; it changes a 1 to a 0 and a 0 to a 1 instead of duplicating them (this change occurs usually with a very low probability).

Part I

Fundamental Theoretical Framework

Chapter 2

Plant Modelling and the Language XL

2.1 Functional-Structural Plant Models (FSPM)

Functional-Structural Plant Models (FSPM) or virtual plant models are models that obviously depicts the development process of plant architecture based on its physiological processes. Here, combining both aspects structural and physiological in modelling becomes reasonable to be realized [170]. According to the triangle of plant models (Figure 2.1; [94]), FSPM represents a powerful combination of approaches for plant modelling. FSPM successfully combines three types of models: structural (morphology), process-based (physiology), and aggregated (statistics). In plant modelling, these three model types are very important. Process models can represent a broad scope of functional aspects of plants, ranging from biophysics and molecular genetics to environmental physiology and agronomy [66]. The development of structural organs of plants intensely relates to their physiological aspects.

Structural models represent plant architecture in 3-d space. In connection with process-based models, they help to realize several concrete ideas, such as to quantify the light interception, calculate some aspects of plant mechanics, understand the plant developmental dynamics, understand some aspects about plant hydraulic architecture, etc. [94]. In addition, statistical models (aggregated models) help to upscale the quantified data from organs to whole plants and plant stands. This is the strong reason why aggregated models are necessary as well to model plants.

Figure 2.2 tells the structure of an integrated single-plant functional-structural model. It describes the tight correlation between morphology and process aspects in plants.

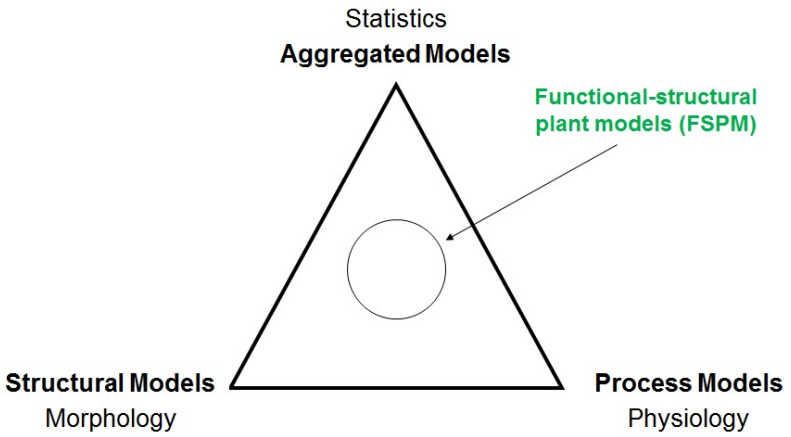

FIGURE 2.1: Triangle of plant models [94]

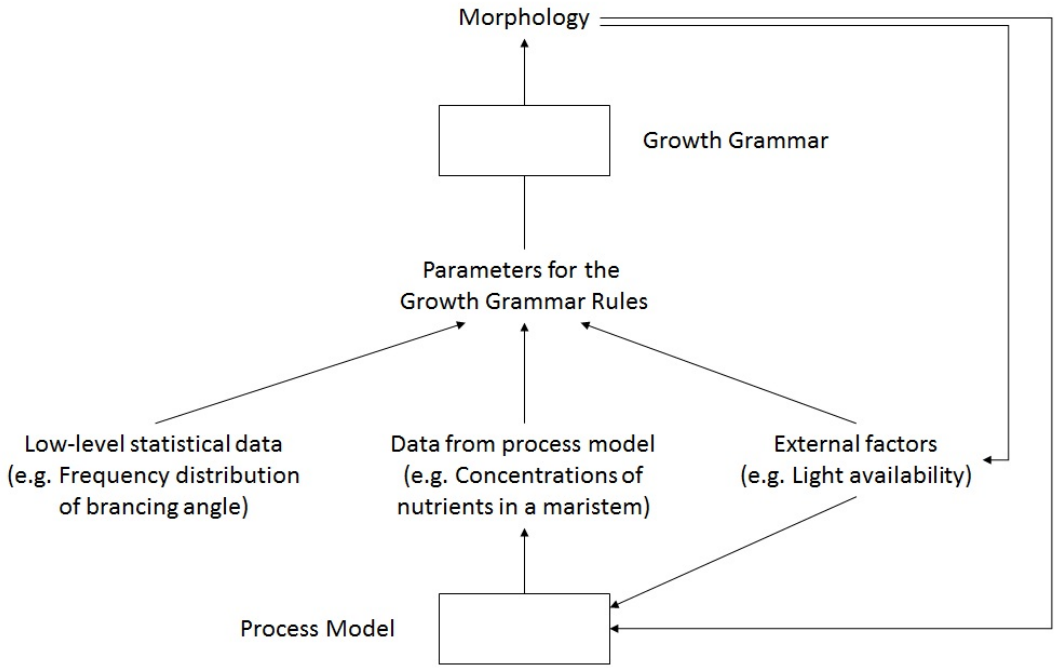

FIGURE 2.2: Structure of integrated single-plant model [94]

Based on a process model, several data describing physiological aspects in the plant can be produced. By combining with statistical data and external factors, data from the process model are used to become parameters for the growth grammar rules. The parameters, by using a growth grammar approach, are used to control a morphology model; the 3-d structure produced by it will, in turn, modify the external factors, e.g. by changing the distribution of photosythetically-active radiation, and will also have a feedback effect on the processes represented in the process model, e.g. by changing the length of transport pathways within the plant. One widely-used technique to realize the structural component of FSPM are rewriting systems, particularly, L-systems. The explanation of the technique will be delivered in the next part.

2.2 Rewriting systems

As mentioned before, L-systems represent a rewriting technique to define a complex object by sequentially interchanging parts, starting with a simple initial object (ω), by means of productions (p) [132], also called rules. A simple example of an L-system is given below.
$\omega : b$
$p1 : a \rightarrow ab$
$p2 : b \rightarrow a$

The example simply expresses that the initial object (ω) is b, it is replaced by using a production (p), $b \rightarrow a$. In the second step, a is replaced by ab using the production $a \rightarrow ab$. The word ab consists of two letters, both of which are simultaneously replaced in the next derivation step. Thus, a is replaced by ab, b is replaced by a, and the string aba is the result. In a similar way, the string aba yields $abaab$ which in turn yields $abaababa$, and so on. The derivation structure of the L-systems example above can be obviously figured by Figure 2.3. For specific purposes, for example graphical modelling, L-systems need a method of implementation. One such method is the turtle concept. It will be explained in more detail in the next section.

2.3 Turtle Concept

One method which can be used in interpreting L-systems for graphical purposes is turtle geometry [2]. A state of the turtle in the plane is defined as a triplet (x, y, α), where the Cartesian coordinates (x, y) represent the turtle's position, and the angle α (heading), measured in clockwise orientation, is represented as the direction in which the turtle is

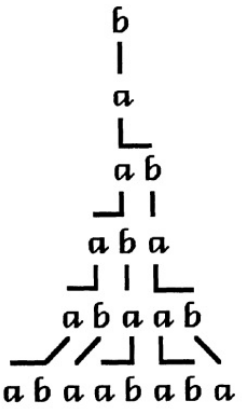

FIGURE 2.3: The derivation structure of a simple example of an L-system [132]

facing. In addition, RU(δ) is a symbol used to depict "turn right" by an angle δ of the turtle, and RU(-δ) is a symbol used do depict "turn left" by δ. The new state of the turtle after execution of the rotation is (x, y, $\alpha + \delta$) or (x, y, $\alpha - \delta$). Furthermore, the symbol "$F0$" is used to move a step of length d with drawing a line, then the state of the turtle changes to (x', y', α), where y'=y+ $d\cos\alpha$ and x'=x+ $d\sin\alpha$; and "$M0$" is a symbol used to move forward a step of length d without drawing a line [132]. The turtle interpretation symbols F, $RU(\delta)$, and $RU(-\delta)$ can be illustrated by using Figure 2.4. For example, a turtle string '$F0F0F0$ $RU(\delta)$ $F0F0$ $RU(\delta)$ $F0$ $RU(\delta)$ $F0$ $RU(-\delta)$ $F0$ $RU(-\delta)$ $F0F0$ $RU(\delta)$ $F0$ $RU(\delta)$ $F0F0F0$' produces the pattern shown in Figure 2.5.

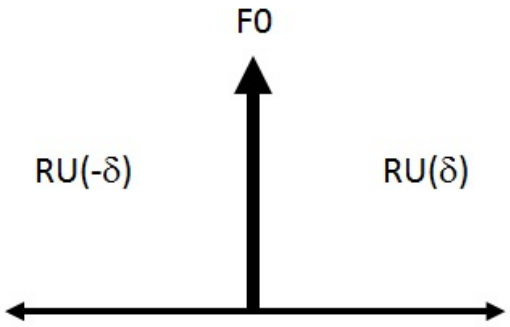

FIGURE 2.4: The interpretation of turtle commands $F0$, $RU(\delta)$, and $RU(-\delta)$ [132]

Turtle interpretation of strings can be extended to 3-d. The key concept is to represent the current orientation of the turtle in space by three vectors H, L, U (see Figure 2.6); indicating the turtle's heading, the direction to the left, and the direction up. In addition, L-systems can be extended in order to model the development of branching structures by putting turtle states on a stack. Especially to delimit a branch in plant model development, there are two symbols used: symbol "[", to push the current state of the

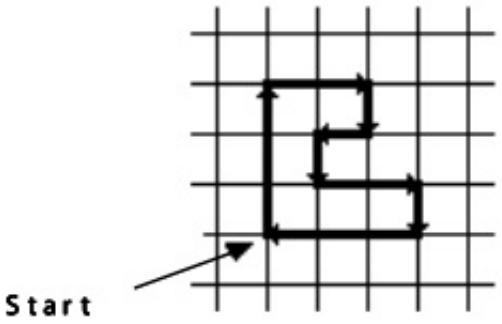

FIGURE 2.5: The interpretation of a string of turtle commands; see text [132]

turtle onto a pushdown stack; and symbol "]", to pop a state from the stack and make it the current state of the turtle.

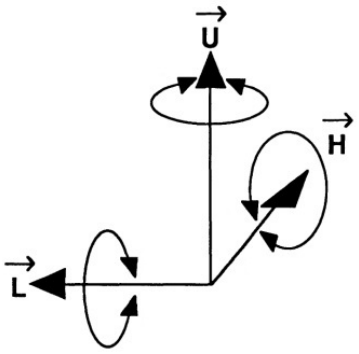

FIGURE 2.6: The turtle orientation in three dimensions [132]

Turtle interpretation of L-systems also can model the natural diversity of plant forms by making some variations in the model. The variations can be achieved by randomizing the turtle interpretation, the L-systems, or both. The implementation of stochastic application of L-system productions will be useful to introduce randomness; such an L-system is a called stochastic L-system. Stochastic L-systems are defined as quadruplets G = (V, ω, P, π), where the function π is a probability distribution to map the set of productions P into a set of production probabilities. A simple example of a stochastic L-system is given below, where each production can be selected with approximately the same probability of 1/3.

$\omega : F0$

$p1 : F0 \xrightarrow{.33} F0[RU(-\delta)F0]F0[RU(\delta)F0]F0$

$p2 : F0 \xrightarrow{.33} F0[RU(-\delta)F0]F0$

$p2 : F0 \xrightarrow{.33} F0[RU(\delta)F0]F0$

2.4 Basic XL Programming with GroIMP

Regarding programming paradigms, the programming language XL supports three paradigms, namely the imperative, object-oriented, and rule based programming paradigms. The imperative programming (also called procedural) consists of explicit commands and procedures to be executed. The commands (instructions) usually manipulate the content of memory cells; they also can call procedures to do several processes, also in procedures, some commands are used. Actually, it is the oldest and most traditional paradigm of programming [16].

In addition, the object-oriented programming is a paradigm that uses the object concept as an interpretation of data structure, procedure or function. The computer is regarded as an environment for virtual objects, which are created and destroyed at runtime of the program. Technically, this paradigm uses object-oriented principles, such as class abstraction, inheritance, polymorphism, and encapsulation [16].

The first two paradigms, indeed, are the basis of the programming language Java [178]. Officially, Java programming uses commands and procedures; however, it also operates with classes, based on the object-oriented principle.

However, Java programming does not have rule-based or graph-related features. To fill this gap, an implementation of relational growth grammars (RGG) was realized [84]. So, the XL language actually uses three programming paradigms: imperative, object-oriented, and rule-based (Figure 2.7).

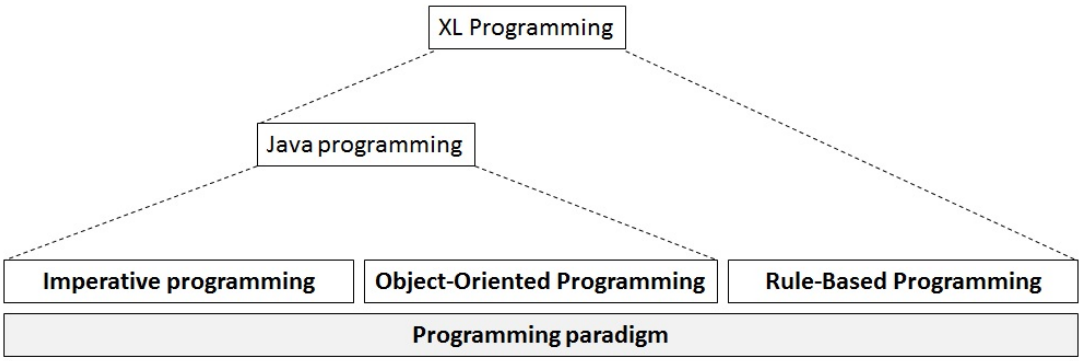

FIGURE 2.7: The paradigms supported in the language XL [90]

Like in other programming languages, the used classes, here called modules, have to be defined in the header of a program. In XL, the module definition follows the well-known concept of object-oriented programming. Four examples of module definition can be seen in Listing (2.1); where A, B, C, and D are the names of modules. The parameters

can be declared between brackets ().

```
module A(float length);
module B(super.length, super.diameter) extends F;
module C (super.length, super.radius) extends Cylinder(length,
   radius);
module D(float length, float radius);
```

LISTING 2.1: Examples of module definitions in XL

Concerning graphical programming, XL provides classes that represent geometrical shapes based on turtle commands [84]. For example the *module* $B()$ and *module* $C()$ in listing 2.1 extend the turtle command F and the standard geometry class constructor $Cylinder()$ (with own parameters). The *module* $B()$ can use a specified list of properties (length, diameter) to identify several parameters from F (inheritance in the object oriented concept) as superclass. Also with *module* $C()$, it can use some properties that belong to its superclass $Cylinder()$; where the superclass can have specific initial values of its parameters. Furthermore, an XL program usually has methods *init*() and *run*() (see Listing 2.2 as example), each of them requiring a specific access key (public, private, or protected). The method *init*() is automatically called when the XL program is executed. The method *run*() is used to contain the productions of the rule-based part. Also, in each rule-based part, common Java programming statements and commands technically can be used and executed by defining them in braces {...}.

```
protected void init()
[
   { <java statements...> }
   Axiom ==> A;
]
public void run()
[
   A ==> B C A;
]
```

LISTING 2.2: An example of methods *init*() and *run*() in XL

Turtle commands can be used in rules of XL, like in L-systems. They can be realized to construct the 2-d or 3-d object structure model. Several basic commands of XL that relate to 2-d and 3-d turtle geometry are described by Table 2.1 [91].

An example of a complete simple program employing turtle commands in XL can be seen in Listing 2.3. It consists only of the method *init*(). The execution result of this code is presented by Figure 2.8.

XL turtle commands also can be realized for specific purposes, such as branching (particularly, this is very important for plant modelling). The notation "[" is used to start

Command	Meaning
F0	'Forward' with construction of an element (line segment, internode, etc.)
F(l)	'Forward' with construction of an element with length l
F(l, d)	'Forward' with construction of an element with length l and diameter d
M0	'Move', forward without construction
M(x)	'Move', forward of length x without construction
P(c)	change current color of object to c ($c = 0..15$)
L(x)	change current step size (length) to x
LAdd(x)	increment the current step size (length) by x
LMul(x)	multiply the current step size (length) by x
D(x)	change current diameter (thickness) to x
DAdd(x)	increment the current diameter (thickness) by x
DMul(x)	multiply the current diameter (thickness) by x
RU(x)	rotation of the turtle around the 'up' axis by x
RL(x)	rotation of the turtle around the 'left' axis by x
RH(x)	rotation of the turtle around the 'head' axis by x
RV(x)	rotation to the ground with strength given by x
RG	rotation absolutely to the ground (direction (0, 0, -1))

TABLE 2.1: Basic turtle commands in the language XL

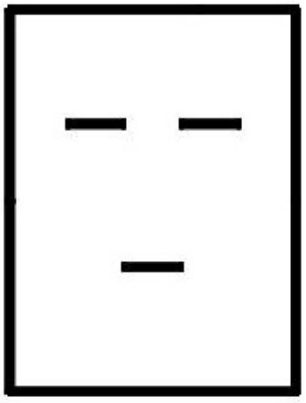

FIGURE 2.8: Executed result from Listing 2.3

the branch, and "]" to define the end of the branch. An example is presented by the simple XL program in Listing 2.4. Easily understood, when firstly it is executed, it will produce only one segment as consequence of the command $F(x)$ (Figure 2.9 A). In the second run, the branch parts will be executed (two instances of branch), and the result will be like in Figure (2.9 B). As it is an iterative code (the right-hand side of the run rule calls the *module* $A()$ itself inside, with a decremented value for parameter x controlling length); when it is iteratively executed many times (for example 7 times), the result will be like in Figure (2.9 C).

```
protected void init()
[
   Axiom ==>   L(100) D(3) F0 RU(90) D(3) F0 F(50)
               D(3) RU(90) F0 F0 RU(90) F(150)
               RU(90) F0 M(40) RU(90) M(30) F(30)
               M(30) F(30) RU(90) M(75) RU(90) M(30) F(30);
]
```

LISTING 2.3: An example of 2-d turtle commands as an XL program

```
module A(float len);

protected void init ()
[
   Axiom ==> A(1);
]

public void run ()
[
   A(x) ==>
      P(0) F(x) [RU(30) RH(90) A(x*0.8)]
      [RU(-30) RH(90) A(x*0.8)];
]
```

LISTING 2.4: An example of 2-d branching generated by an L-system in the XL code

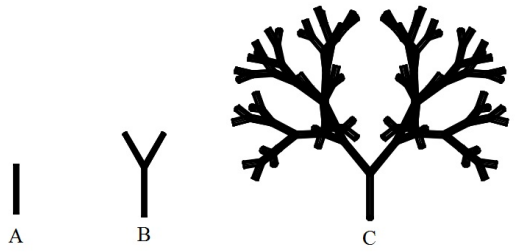

FIGURE 2.9: Executed result from Listing 2.4

Another example of XL code comes from [22]. It describes a simple virtual plant with branches and leaves. The program consists of four modules and two methods. *Module Bud*() and *module Node* extend the superclass *Sphere*, *module Internode*() extends the turtle command F, and *module Leaf*() extends the superclass *Parallelogram*(). Every *Internode* and *Node* are followed by *Bud* and *Leaf* with specific angle of vector L (divergence angle of branching). The program execution result can be seen in Figure 2.10.

```
module Bud extends Sphere(0.1)
{
   {setShader(RED);}
}

module Node extends Sphere(0.07)
{
   {setShader(GREEN);}
}

module Internode extends F;
module Leaf extends Parallelogram(2,1);
```

```
//simple plant, with leaves and branches:
protected void init()
[
   Axiom ==> Bud;
]

public void run()
[
   Bud ==> Internode Node [ RL(50) Bud ] [ RL(70) Leaf ]
      RH(137.5) Internode Bud;
]
```

LISTING 2.5: Other example of XL code. It describes a simple plant with branches and leaves

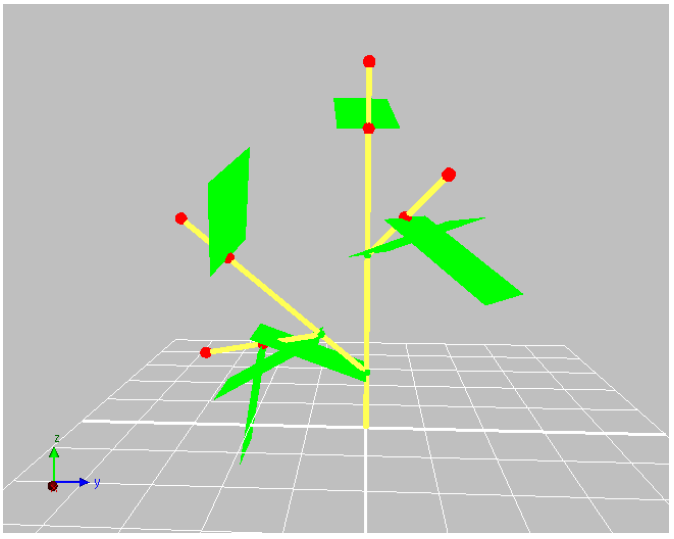

FIGURE 2.10: Executed result from Listing 2.5 [22]

Command	Meaning
$int\ i;$	declaration of an integer variable with name i
$float\ a = 0.0;$	declaration / initialization of a floating point variable
$int[]\ x = new\ int[20];$	declaration of an array of length 20 ($x[0]$, ..., $x[19]$)
$float[]\ y = \{0.1, 0.2, -1.4\};$	declaration and initialization of an array
$i = 25;$	assignment
$i++;$	i is incremented by 1
$i--;$	i is decremented by 1
$i+ = 5;$	i is incremented by 5
$i- = 5;$	i is decremented by 5
$i* = 2;$	i is doubled
$i/ = 3;$	i gets the value i/3
$x = Math.sqrt(2);$	x gets assigned the square root of 2
$if\ (x! = 0)\ \{y = 1/x;\}$	conditional assignment of $1/x$ to y
$while\ (i <= 10)\ \{i++;\}$	loop: as long as $i \leq 10$, i is incremented by 1
$for\ (i = 0;\ i < 5;\ i++)\ \{i+ = 5;\}$	imperative for-loop
$if\ (i == 0)\ \{...\}$	test for equality (" = " would be assignment!)

TABLE 2.2: Examples of imperative code in XL

Table 2.2 shows some examples of imperative code in XL. The commands like assignment of values to variables, arithmetical operations, function calls, output (print commands), etc. technically are specified in the same way like in Java code and are enclosed in braces {...} [92].

Furthermore, some features of XL that are used directly in this research will be explained here, such as $NURBSSurface$, instantiation rules, etc. $NURBSSurface$ is one XL feature that relates to complex 3-d geometry. It has an attribute of type $BSplineSurface$ or $BSplineCurve$. They are respectively used to render the surface or curve of the object in 3-d using the framework of non-uniform rational *B-Spline* [83]. In this research, the attribute input used is $BSplineSurface$. A possible construction of a $NURBSSurface$ can be seen by an example in Listing 2.6. The leaf surface itself is defined as class $ExtrudedSurface$ by using the method $BSplineOfVertice$ that relates to the profile ($profile$) and trajectory ($traj$) of a leaf. The parameter $profile$ defines the profile of the leaf through 10 parameters, and the parameter $traj$ defines the trajectory of the leaf through 12 parameters. The simulation output of Listing 2.6 can be seen in Figure 2.11.

```
protected void init ()
[
   Axiom ==>
   {
      VertexList profile = new VertexListImpl (new float[]
      {-1, 0, -0.5, 0.5, 0, 0, 0.5, 0.5, 1, 0}, 2);
      VertexList traj = new VertexListImpl (new float[]
      {0, 0, 0, 0.1, 0, 0, 1, 1, 0, -1, 2, 0.01}, 4);
   }
   NURBSSurface(
      new ExtrudedSurface(
         new BSplineOfVertices(profile, 3, false, false),
         new BSplineOfVertices(traj, 3, false, false)
      ).(setUseScale(true))
   ).(setShader(new RGBAShader(0.2,0.4,0)));
]
```

Listing 2.6: Format of NURBSSurface declaration in XL

Another feature of XL is the instantiation rule. It is a possibility of declaring a module that refers to the multiple inclusion of the same geometric structure but with different location in space [83]. The module declaration principally can be called recursively and with references to graph nodes. It enables an instancing of graphs at runtime for frequently occurring sub-structures. It helps to minimize the computer memory usage during simulation [148]. Generally, it is used to create a complex geometry out of a compact specification. An instantiation rule consists of a module declaration on the left-hand side and contains the production statement of the instantiation rule on the right-hand side. An example of instantiation rule declaration can be shown by Listing 2.7; by using parameter n, the algorithm instantiates a sequence of $2n$ cylinders with alternating colors [83].

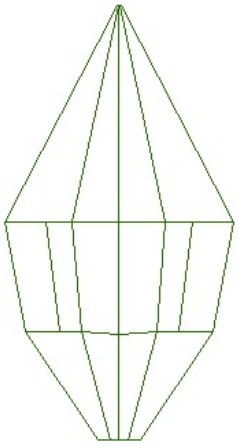

FIGURE 2.11: Executed result from Listing 2.6 [56]

```
module Y(int n)
==>   for(1 : n)(Cylinder.(setShader(RED))
      Cylinder.(setShader(GREEN))
```

LISTING 2.7: Example of instantiation rule declaration

Regarding the radiation model, XL provides four types of light source (*DirectionalLight*, *PointLight*, *SpotLight*, and sky / sun sky shader) [154]. A light source is needed to compute the amount of radiation. The radiation model itself is made accessible by the class *LightModel* with a declaration like in Listing 2.8; where *lm* is a variable representing the Light Model, *nRay* is the number of random rays, and *nRef* is the recursion depth or number of reflections.

```
LightModel lm = new LightModel(nRay, nRef)
```

LISTING 2.8: XL code for light model declaration

Furthermore, the execution arrow (::>) represents a single imperative statement or a list of imperative statements enclosed by braces {...} that are located in its right-hand side [83]. It executes the statement in every match of the left-hand side. It is useful if modification of the topology of the current graph is not desired, but only of its internal state, i.e., attribute values of nodes.

Chapter 3

Fundamental Facts about the Rice Plant

3.1 The Life Cycle of Rice Plant

The life cycle of a rice plant lasts generally 100-201 days. The varieties with growth duration around 130-150 days are usually photoperiod sensitive and planted in deep water area [167]. Rice is a very flexible plant. It can grow well under both flooded and rainfed conditions [180]. Fundamentally, there are three phases of rice plant growth: vegetative, reproductive, and ripening phases [183] [167] [104]. Even though several researchers categorize the life story of a rice plant into two growth periods: vegetative and generative period [108]; both theories talk about the same stages in detail (see Figure 3.1). The rice plant consumes around 35 and 30 days to grow in reproductive and ripening phases respectively; however, especially for the vegetative phase, the time consumed by one rice plant to grow depends on the variety (see Figure 3.2). This causes differences in growth duration (plant age) of rice plants.

Regarding the time period spent in the vegetative phase, the rice plant varieties are grouped into three groups. The first comprises early-maturing varieties. They initiate a panicle primordium before reaching the maximum tiller number (type A in Figure 3.2). The second one comprises the varieties with adequate vegetative phase. They initiate panicle primordium right after the maximum tiller number is reached (Type B in Figure 3.2). The third one comprises the late-maturing varieties. They have long periods of vegetative growth and reach the maximum tiller number condition before initiation of panicle primordia (type C in Figure 3.2) [156].

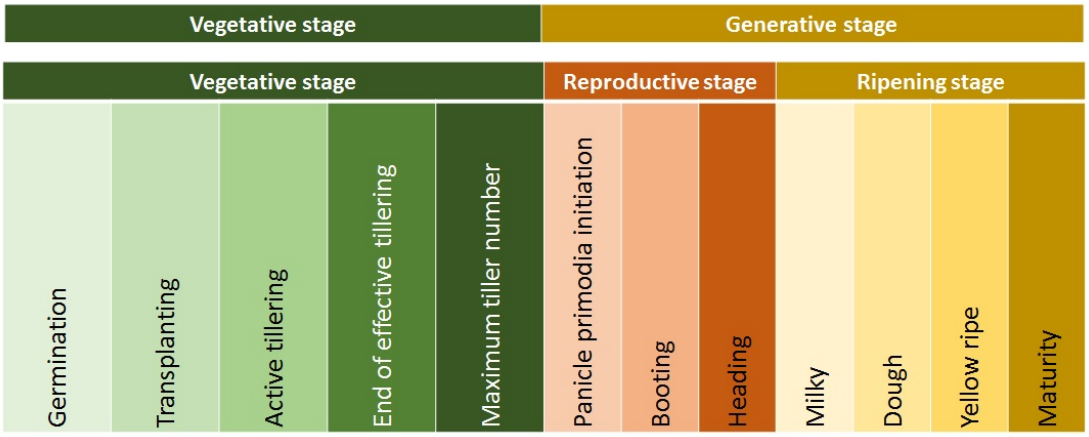

FIGURE 3.1: Two types of rice plant life cycle [108] [183] [167] [104]

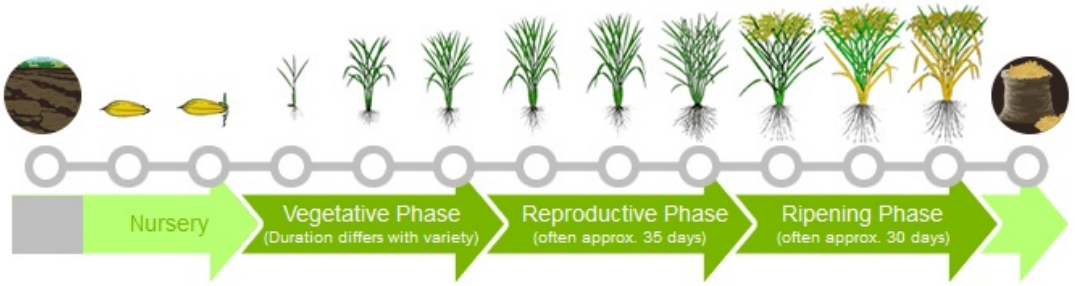

FIGURE 3.2: The period of rice plant growth [71]

As mentioned before, especially for the vegetative phase, it is characterized by active tillering, plant height increasing, and regular leaf emergence. All organs contribute to increasing the leaf area in intercepting the sunlight [183]. The vegetative phase also can be called the growth period of stems and leaves [108]. Tillering starts when the main stem has 4-5 leaves. It can be seen that in the final state, the main stem is usually longer than tillers.

The first three detail stages of the vegetative phase are: germination, seedling, and tillering (which can again be differentiated into active tillering, end of active tillering, and reaching maximum tiller number). In the germination stage (stage 0), the embryo will germinate when it finds sufficient moisture. The germination stage covers the period from emergence of either the coleoptile or the radicle to the emergence of the first leaf. The seedling stage (stage 1) covers the period from the emergence of the first leaf to the emergence of the fifth leaf. The tillering stage (stage 2) starts from the growth of the fifth leaf, the number of tillers increases until maximum tillering [183] [180].

Indeed, the plant height does not increase much in the vegetative phase; the plant height

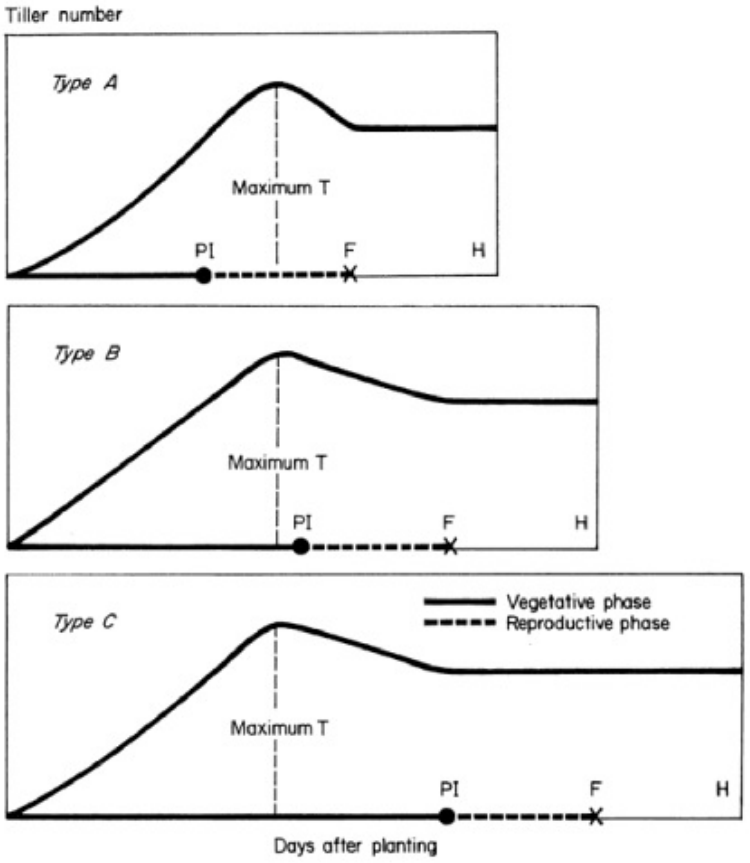

FIGURE 3.3: Three types of varieties based on the duration of the vegetative phase [156]

increases more rapidly in the reproductive phase which is characterized by stem elongation (stem elongation stadium). It is proven, that the stem length in the reproductive phase is longer than the stem length in the vegetative stage. A decline in tiller number, emergence of the flag leaf (the last leaf), booting, heading, and flowering are several characteristics of the reproductive phase of rice plant growth [183].

3.2 Vegetative Green Organs of the Rice Plant

Generally, there are five organ types of rice plants: roots, stems, leaves, and panicles (see Figure 3.4). Especially, talking about vegetative organs, there are three organs: root, stem, and leaf. However, our research only concerns the above-ground rice plant during the vegetative phase, thus the organs stem and leaf will be mentioned in detail here.

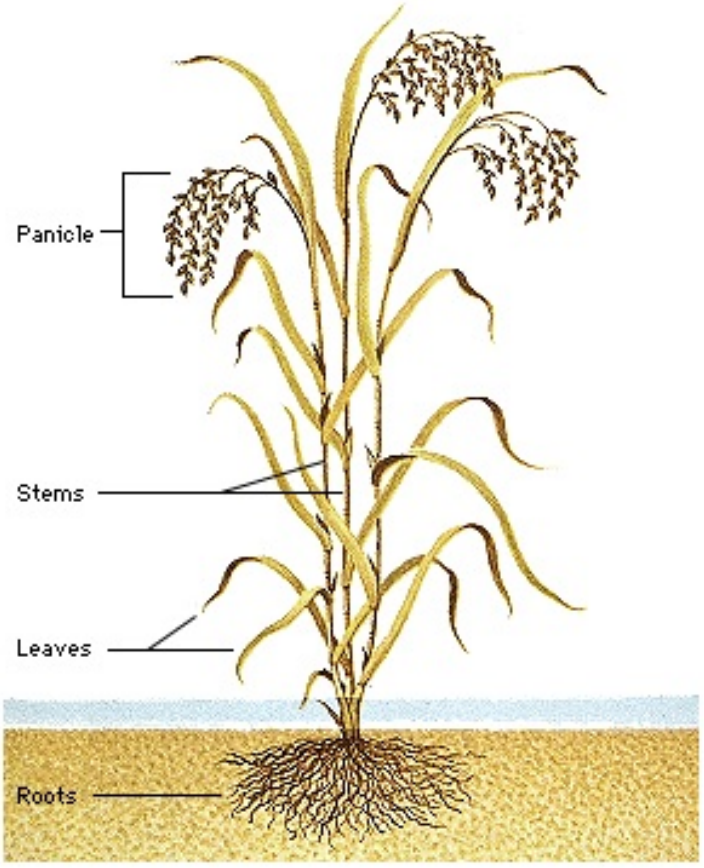

FIGURE 3.4: General parts of rice plant [67]

3.2.1 Main Stem and Tillers

A stem consists of a series of nodes and internodes (see Figure 3.5). Physically, internodes are round and hollow, with smooth surface. The upper internodes are longer than the lower ones [180]. The main stem and tillers have different height. Because, tillering starts when the main stem has 5-6 leaves [183]. Main stem and tillers have usually one internode less than their number of leaves, and primary and secondary tillers have around 12 and 10 internodes respectively [189].

3.2.2 Rice Leaf

Every rice leaf consists of sheath, blade, auricle, and ligule. The auricle and ligule distinguish rice from barnyard grasses [104]. However, several rice varieties lack the auricle and ligule, they are called liguleless rice [183].

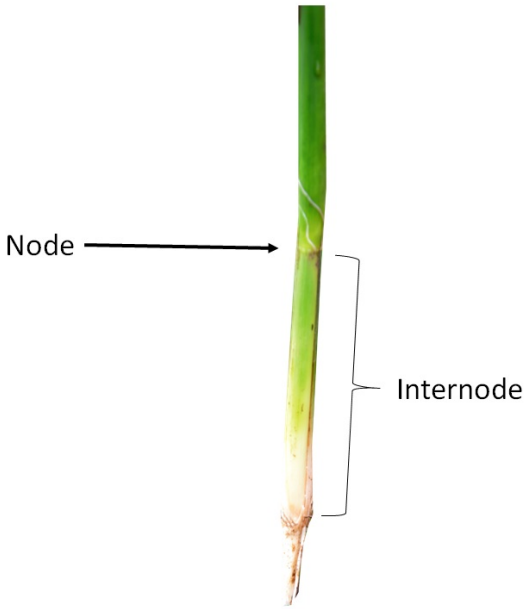

FIGURE 3.5: Single stem of rice

Rice leaves are narrow and flat (see Figure 3.6), they grow alternately on the stem, with one leaf on one node. In fact, leaves come out one after another, and the leafing interval is almost constant under a specified condition. One single leaf needs approximately 4-5 days to fully grow in the early growth phase, and 7-8 days in the next stages. Leaves are the engines of the plant to capture the light and produce carbohydrates. The last leaf is called the flag leaf [156] [183] [115] [104] [180]. There are about 14-18 leaves on the main stem (depending on variety), and usually the 4th leaf from the flag leaf is the longest one [167] [104].

The rice leaf is the green organ that most strongly relates to yield productivity. The leaf is the green organ that must be considered in breeding research with respect to properties such as erectness, length, width, thickness, and color [77]. However, the correlation of leaf thickness and productivity is still not proven yet [183]. The increase of leaf area index (LAI) is affected by the increase of tillers and each leaf area itself, where LAI is "the amount of leaf area (m^2) in a canopy per unit ground area (m^2)" [8]. Some commonly required leaf characteristics are erect, thick, small, and short [104]. A picture of rice leaves can be seen in Figure 3.6.

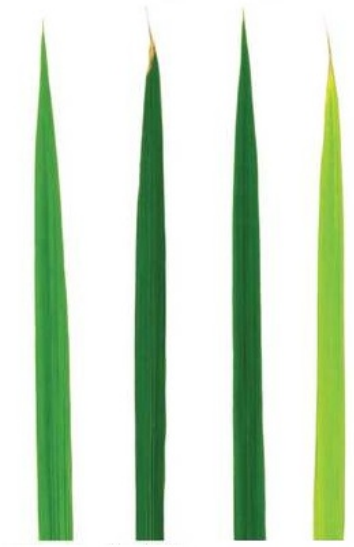

FIGURE 3.6: Simple view of rice leaf [1]

3.3 The Classification of Indonesian Varieties of the Rice Plant

There are four kinds of Indonesian rice plants as the research objects. They are the varieties Dodokan, Fatmawati, INPARI 9 and Ciherang. Based on plant age classifications, the varieties represent three classes. They respectively represent very early, early, medium and early. The varieties Fatmawati and Ciherang represent the same age classification, however they have clearly a different size of leaves. Based on plant age, the rice plants basically are classified into six classes [70]: ultra early (plant age is < 85 days), super early (plant age is 85 – 94 days), very early (plant age is 95 – 104 days), early (plat age is 105 – 124 days), medium (plant age is 125 – 164 days), and late (plant age is > 165 days).

Also, actually the rice plants can be classified by their form. The form describes how wide the angle between main stem and its tillers is (axial angle). The classes are [69]: erect, where the axial angle is $< 30^o$, semi-erect, where the axial angle is $\pm 45^o$, open, where the axial angle is $\pm 60^o$, spreading, where the axial angle is $> 60^o$, and the tillers do not touch the ground, and prostrate, where the axial angle is $> 60^o$, and the tillers touch the ground. Regarding leaf position, there are four classes. The leaf position is the position of the leaf relative to the stem that is measured as the angle between leaf tip and stem (leaf chord angle). The classes are [69]: erect, where the leaf chord angle is $< 45^o$, semi-erect, where the leaf chord angle is between 45^o and 90^o, horizontal, where the leaf chord angle is 90^o, and recurved, where the leaf chord angle is $> 90^o$.

No	Parameter	Dodokan	Fatmawati	Inpari 9	Ciherang
1	Plant age (*days*)	116	122	125	121
2	Age classification	early	early	medium	early
3	Plant height (m)	0.88	1.03	1.13	1.11
4	Height classification	short	short	short	short
5	Plant form	semi-erect	semi-erect	erect	erect
6	Leaf position	semi-erect	erect	erect	erect
7	Maximum leaf length (m)	0.48	0.59	0.49	0.46
8	Maximum leaf width (m)	0.011	0.019	0.013	0.014
9	Leaf number	14	15	17	16
10	Maximum stem length (m)	0.73	0.87	0.81	0.74
11	Maximum stem diameter (m)	0.0022	0.0025	0.0027	0.0029
12	Productive tiller number	10-13	8-14	16-22	14-17

TABLE 3.1: The morphological characteristics of four Indonesian varieties of rice plant (average values) [146]

The rice plants also can be categorized by their height. The height is calculated from the ground until the highest organ part of the plant. There are three simple categories of plants based on height. The categories are [69]: short, where the plant height is $<$ 110 cm; medium, where the plant height is between 110 and 130 cm; and long, where the plant height is $>$ 130 cm.

[69] also classified the leaf length into five categories. They are: very short, where the leaf length is $<$ 21 cm, short, where the leaf length is between 21 – 40 cm, medium, where the leaf length is between 41 – 60 cm, long, where the leaf length is between 61 – 80 cm, and very long, where the leaf length is $>$ 80 cm.

Table (3.1) describes the morphological characteristics of the four Indonesian varieties of rice that are taken as research objects. The empirical data sometimes only informs one of two types of data, quantitative or qualitative data.

Chapter 4

Algorithms for Optimization

4.1 The Optimization

Optimization is a process to find the conditions that give the optimum value of a function. The value can be the minimum or the maximum of the function. The function is called the objective function and has to be formulated before. Mathematically, the minimum value of the function ($f(x)$ for example) actually is same as the maximum value of the negative of the function ($-f(x)$)(see the illustration in Figure 3.1; [133]). The word 'optimum' itself comes from the Latin word 'optimus' meaning best. Thus, optimization can be defined as a way to find the best solution of a problem [48].

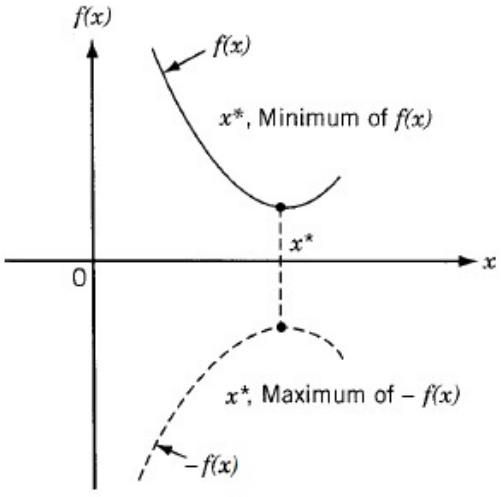

FIGURE 4.1: Illustration of the minimum and maximum value of the function. The minimum value of the function is same as the maximum of the negative of the function [133].

Basically, the objective function has to be defined mathematically to describe the optimization problem that will be solved. The equations (4.1) upto (4.5) are examples to describe how the objective function and some boundary conditions for its arguments are defined. According to the equations, f is an objective function, where the problem is to find x_1 and x_2 (as independent variables). By satisfying equation (4.2) upto (4.5), and inserting them to equation (4.1), x_1 and x_2 can be tested.

$$
\begin{aligned}
Maximize: \quad & f(x_1, x_2) = 2.5x_1 + 0.75x_2 && (4.1)\\
Subject\ to: \quad & x_1 + x_2 \geq 3.6 && (4.2)\\
& 0.6x_1 + 0.4x_2 \geq 1.2 && (4.3)\\
& x_1 \geq 0 && (4.4)\\
& x_2 \geq 0 && (4.5)
\end{aligned}
$$

Indeed, the methods for seeking the optimum value are known as mathematical programming techniques. They commonly are studied as part of operational research studies. The operational research methods can be classified into three classes: mathematical programming or optimization techniques, stochastic process techniques, and statistical methods. The mathematical programming techniques are fruitful for finding the optimal value of an objective function, particularly in the linear case, like in equations (4.1) upto (4.5) above. The stochastic process techniques are useful to find the optimal value for a problem that is described by a set of random variables. And the statistical methods are useful to analyze experimental data and to develop empirical models to find the most accurate representation of the physical situation. Briefly, the classes of operational research methods with some method examples are listed in Table 4.1 [133].

Other optimization methods can be called modern or non-traditional optimization methods. Several examples are: genetic algorithm, simulated annealing, ant colony optimization, swarm optimization, neural networks, and fuzzy optimizations [133]. They are heuristical methods, i.e., they usually do not five the exact global optimum of a problem, but an approximation. On the other hand, they are often faster than exact methods, particularly in the case of non-linear objective functions with many variables.

Several statistical methods for sampling also can be used as optimization methods. Such statistical sampling methods will be delivered in the next sections, they are full-factorial design, simple-random sampling, and latin-hypercube sampling. Two types of non-traditional optimization methods (hill-climbing and simulated-annealing) will be mentioned as well. Five types of optimization methods will be used in our study of light

Mathematical programming or optimization techniques
Calculus method
Linear programming
Non-linear programming
Stochastic programming
Multi-objective programming
Game theory
Stochastic process techniques
Statistical decision theory
Markov process
Simulation methods
Statistical methods
Regression analysis
Cluster analysis, pattern recognition
Design of experiments

TABLE 4.1: Methods of the operational research [133]

interception in rice. Their following description will partially be based on the overview given in [157].

4.2 Full Factorial Design

In statistics, especially in the design of experiment (DoE), a full factorial design (FFD) that was proposed by [46] is a factorial experiment design which ensures that all possible factor level combinations (i.e., combinations of values of the variables of the objective function) are defined and used. By this technique, all group differences can be explored. A full-factorial experiment design actually consists of every combination of the levels of factors in the experiment [111]. In other words, FFD is an experiment design in which every possible combination of factor levels (values of variables) is tested [97]. It means, for example, the possible combinations will be 8 possibilities for 3 factors or variables with two levels for each factor or variable (2^3). Practically, it will be more complex; if FFD must be implemented in several factors or variables (k) with different levels (l) for each factor or variable. The total combination possibilities (CP_{tot}) can be calculated by equation (4.6).

$$CP_{tot} = \Pi_{k=1}^{n} l_k \tag{4.6}$$

The disadvantage of this method is that FFD can take a lot of time and money, because it includes all possible combinations of every level of every factor and can require a lot of trials. It is not appropriate for an optimization process with a big number of samples or

combinations. However, regarding optimization, this approach surely can get the global optimum value if the granularity of the factor levels is fine enough. The implementation of this approach in optimization can be described by the simple pseudocode of the basic procedure in Listing 4.1.

```
Procedure FullFactorialDesign(prodedureParameters)
Begin
   <...variables definition...>

   //initiating starting point
   para <-- lowBo
   bestVal <-- 0

   //full factorial design looping
   While(maxVal=false)
      //getting value of objective function
      //and comparing with the best
      curVal <-- objFunction(para)
      If(curVal > bestVal)
         bestVal <-- curVal
      End if

      para <-- nextIndexPara

      //checking all parameters must be less than upper bound
      maxVal <-- maxValCheck(para)
   End while

   //for last time, getting value of objective function
   //and comparing with the best
   curVal <-- objFunction(para)
   If(curVal > bestVal)
      bestVal <-- curVal
   End if
End
```

LISTING 4.1: Pseudocode of basic procedure for the full factorial design method

The aim of the FFD procedure in Listing 4.1 is to find the global optimum value (the highest or lowest value) that can result from the objective function ($objFunction()$), where several parameters or parameter combinations ($para$) are its parameter inputs. Simply, the procedure must check one by one all parameter combinations, starting from the first parameter combination upto the last one, by inputting the parameters into the objective function.

The parameters are defined with a value of lower boundary ($lowBo$) as the first parameter combination, and variable $bestVal$ is defined as 0. Then, the procedure inputs the parameters into the objective function, where the value of the objective function is hold in the variable $curVal$. Next, the value of $curVal$ is compared with $bestVal$; if it is higher than $bestVal$, it is set as $bestVal$ then. The process is continued with the next parameter combination, to get the next value of the objective function, until the last one (where the boolean $maxVal$ is true). The global optimum can be reached after the procedure scanned all possible values of the objective function.

Be concluded here, that the specific parameters used in the FFD procedure could be several. They are: $curVal$, $bestVal$, $nextIndexPara$ and $maxVal$. They respectively represent a current value, it is used to save the value of objective function; a best value, it is used to save the best value after comparing with the next value of objective function; a index for next parameters, it is used to move to the next parameter combination, and a maximum value, it is used to state the status "true" that the repetition process has reached the upper bound.

4.3 Simple Random Sampling

Simple random sampling (SRS) was proposed since 1887 by [80] in [112]. Indeed, SRS is one of several methods for population-based survey. It is a basic selection process of sampling. The subjects in the population are sampled randomly. The process uses a random number generator or a random number table, so a same results from several random sampling processes will be technically avoided [53]. SRS is the simplest sampling method. It independently generates each sample element of supposedly uncorrelated variables. In SRS, every member of the population has the same chance and probability to be selected. This method is appropriate for a small number population, it is not convenient for a large population [110] [63].

Figure 4.2A is an example of sampling results produced by using the SRS method. It is an example of sampling for two dimensional (two variables) combinations. The sample generated from eight random choices can turn out to cover the space of all value combinations unevenly. In the example, combinations of high values of variables A and B are under-represented in the sample. Moreover, the same result can be obtained from independent sampling. For the optimization process, the process to take samples randomly can be applied for choosing the objective function argument values. Here, Listing 4.2 describes the basic procedure of simple random sampling as an optimization process.

```
Procedure SimpleRandomSampling(procedureParameters)
Begin
   <...variables definition...>
   Make parameterTable
   bestVal <-- 0

   While(loopCount)
      //randomizing new parameter combination and checking the
      //equality
      randLoop <-- true
      While(randLoop=true)
         //randomizing new parameters or parameter combinations
         para <-- random()

         //checking the equality
         If(para in parameterTable)//finding equality
            randLoop <-- true
```

```
         Else
            randLoop <-- false
         End if
      End while

      //getting value from objective function
      curVal <-- objFunction(para)

      //checking the best value
      If(curVal > bestVal)
         bestVal <-- cVal
      End if

      //updating parameter table
      put para in parameterTable
   End while
End
```

LISTING 4.2: Pseudocode of basic procedure for simple random sampling without repetitions of identical parameter value combinations

The goal of SRS in Listing 4.2 is to find the global optimum value (the highest or lowest) of the objective function in some tests. How many times testing will be conducted can be defined in the variable *loopCount*. And to check the similarity of parameter combinations, a *parameterTable* is defined. It means, the tests that are conducted will not repeat the same combination of parameter values twice.

Different to FFD, where parameter combinations are checked one by one; in SRS, the parameter combinations are selected randomly in every time of test. The probability to get the global optimum value strongly depends on how many times the test is conducted. The probability to get the global optimum value is small for a big population (many parameter combinations).

For the big possibilities, to get the same parameter combination is one thing nearly impossible. However, making the historical table (called above a random number table) is one way to avoid the same random value. To realize the purpose, the parameter *curentPositionHistory* is needed. It is technically used to record the parameter combinations that have been used before. The parameters *loopCount*, $curVal$, and $bestVal$ are required. They are respectively used set the number of looping, to save the current value, and the best value.

4.4 Latin Hypercube Sampling

Latin hypercube sampling (LHS) was introduced by [109]. Like the methods before, it is a searching technique in n-dimensional space [35] [5]. It is a method to take a sample from a population that can produce input values to estimate output variables of functions. Its variance of samples is less than that obtained by the simple random

sampling method [151]. Indeed, it is like simple random sampling, as a probabilistic sampling in the sense that a weight can be associated with each sample that can be used in probabilistic calculation. However, it emphasizes the uniform sampling of the distributions [40] and practically it can produce more stable analysis outcomes than simple random sampling does [63], as in LHS, the input variable value are divided into several predefined cubes (clusters).

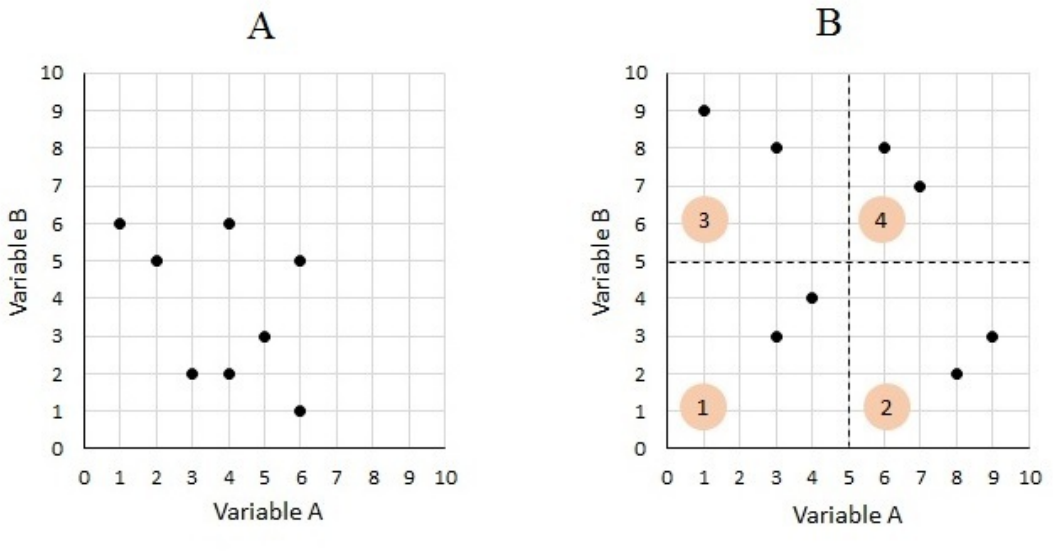

FIGURE 4.2: Simple example of sampling in a 2-d space of value combination. (A) Samples resulting from the method of simple random sampling. (B) Samples resulting from the method of latin hypercube sampling with four clusters.

A simple example of an LHS result can be seen in Figure 4.2B. The result is more grouped than in the case of SRS. By using four predefined cubes (clusters) with maximum two sample members in one cube, eight realization of sampling successfully group the variable combinations into that four groups. In addition, the basic procedure of LHS can be depicted by the pseudocode in Listing 4.3.

The intention of LHS optimization is similar to the two previous methods (FFD and SRS), it is used to get the optimum value of the stated objective function. Essentially, LHS is more similar to SRS, it randomizes parameters that will be used to produce the values of the objective function. However, in LHS, parameter values are grouped into several specified clusters (cubes).

In every parameter combination that is randomly chosen, it is checked, which cluster the parameter combination is addressed in (function *checkCluster*()). Consequently, how many parameter combinations can be addressed in one cluster should be defined first.

Several specific parameters are required in LHS method. The parameter *loopCount* is a numeric parameter. It is used to set, how many times the looping process is conducted. The parameter *loopCount* is $oneClusMaxNum \times clusNum$; where $oneClusMaxNum$ is a maximum number of how many times one cluster is visited by the randomized parameter combination, and $clusNum$ is a number of cluster. The parameter $clusNum$ itself is a total number of multiplication of all *clusPart* number; where *clusPart* is a cluster part, it means one parameter can possibly divided into several parts.

```
Procedure LatinHypercubeSampling(procedureParameters)
Begin
   <...variables definition...>
   <...parameters cluster making...>

   bestVal <-- 0

   While(loopCount)
      //randomizing new parameter combination and checking the
      //equality
      eqStatus <-- true

      While(eqStatus=true)
      //randomizing new parameter
         para <-- random()

         //checking cluster equality status
         eqStatus <-- checkCluster(para, clusterBoundary)
      End while

      curVal <-- objFunction(para)

      //checking the best value
      If(curVal > bestVal)
         bestVal <-- cVal
      End if

      loopCount++
   End while
End
```

Listing 4.3: Pseudocode of basic procedure for latin hypercube sampling method

Other specific parameters are *clusLowBo*, *clusUpBo*, and *clusStatus*. They respectively are the parameters lower bound of cluster, upper bound of cluster, and cluster status. Especially for cluster status, it is used to set a number of how many times one cluster has been visited by the parameter combination. To count the cluster number, the parameter *statusCounter* is needed. In addition, the parameters *intervalPoint* and *eqStatus* are also required. They are respectively used to be a cluster pointer and an equality status of cluster.

4.5 Hill Climbing Method

The hill climbing (HC) method is a local search method. It was proposed by [113]. Simply stated, it contains a loop that continually moves in one direction of an increasing value (uphill), and never goes downhill. It is terminated when it reaches a 'peak', where there is no neighbor having a higher value. The probability is high that it will be trapped in one local maximum (one hill). A local maximum is a peak that is higher than its neighbors but lower than the global maximum [138]. The basic procedure of the HC method is presented in Listing 4.4.

```
Procedure HillClimbing(procedureParameters)
Begin
   <...variables definition...>

   //randomizing new parameter combination
   para <-- random()
   curVal <-- objFunction(para)

   //looping until local optimum is found
   While(search is not terminated)
      //finding the best neighbor
      bestNeighbor <-- getBestNeigh()

      //finding local optimum
      If(curVal>=bestNeighbor)
         Local optimum is found
         Search is terminated
      Else //move to next position
         curVal <-- bestNeighbor
      End if
   End while
End
```

LISTING 4.4: Pseudocode of basic procedure for hill climbing method

In Listing 4.4, it is obviously mentioned that a looping process for seeking the optimum value can be terminated when the local optimum value is found. This occurs when the variable $curVal$ is better than the value of the variable $bestNeighbor$. This means, no parameter combination that belongs to its neighbors can give a better value of the objective function than the variable $curVal$.

Several specific parameters required by HC method can be listed here. The parameters $curVal$, $bestNeighbor$, and $bestVal$ are parameters necessitated in this method. They are respectively current value, best value of neighbor, and best value (should be as a local or global best value).

4.6 Simulated Annealing Method

Basically, the simulated annealing (SA) method was proposed by [82] and also by [26]. It is a probabilistic method for obtaining the global minimum of an objective function. Technically, it can possibly deliver several local minima [15]. The idea of SA firstly came from the metallurgy field. The objective function is treated as the energy function of a molten metal exposed to an artificial temperature which is gradually cooled [181]. The temperature controls the probability that the algorithm moves to a neighbor with a worse value of the objective function (as opposed to the HC method when such a move can never happen). The process will terminate when the adjusted temperature reaches the optimal temperature. This scenario can still deliver a local optimum which is not the global optimum, but the algorithm has a chance to escape from this trap as long as the temperature is still high enough. The pseudocode of the basic procedure of the simulated annealing method is given in Listing 4.5.

```
Procedure SimulatedAnnealing(procedureParameters)
Begin
   <...other variables definition...>
   baseTemp <-- 100;
   optTemp <-- 40;

   //randomizing new parameter combination and getting value of
   //objective function
   para <-- random()
   curVal <-- objFunction(para)

   //looping until local optimum is found
   While(baseTemp>=optTemp)
      //finding the best neighbor
      bestNeighbor <-- getBestNeigh()

      //finding local optimum
      If(bestNeighbor>=curVal)
         curVal <-- bestNeighbor
         bestNeighbor <-- 0
      Else
         delta <-- bestNeighbor - curVal
         //Probability
         If(exp(-delta/baseTemp) > random(0..1))
            curVal <-- bestNeighbor
            bestNeighbor <-- 0
         End if
      End if

      //adjusting baseTemp
      adjust(baseTemp)
   End while
End
```

LISTING 4.5: Pseudocode of basic procedure for simulated-annealing method

The variable *delta* is a key concept in the SA method. Because, when $curVal$ is better than $bestNeighbor$, *delta* is used to check the probability to move or to stop by calculating $-delta/baseTemp$ and then comparing to a value $random(0..1)$. This condition

makes SA more flexible than HC. It can be concluded here that the specific parameters used in simulated annealing models are $baseTemp$ and $optTemp$.

Part II

Research Methodology

Chapter 5

Research Methodology

5.1 Research Methodology Structure

All stages of this research are presented by Figure 5.1. Basically, the research uses the rice plant as a research object, optimization as a way to seek optimal rice plant structure among a big number of possibilities, and a skylight model as a developed model to be embedded to see sunlight radiation effects in the light interception process by the plant. A preliminary analysis of the three aspects is the first stage of the research. Testing several simple coded methods, reading various papers, and tracing the state of the art in the research field; are several activities that have been done in this stage. The fundamental comprehensive understanding of research is the goal of this stage.

Furthermore, quantitative and qualitative data from other researchers, also theoretical and practical side; are features that are initially analyzed in the next stage (literature review). [104], [183], [155], [189], and [175]; are studies that relate to the rice plant, especially to its morphological aspect, and have become main sources used to build the model. They provided components for our rice plant model. And more than 40 other papers and books have been reviewed comprehensively.

Particularly, in testing some coded methods, Niklas' studies (in 1994, 1999, 2000, and 2004) [119], [120], [121], and [122]; about plant evolution have partially been implemented in XL on the model platform GroIMP.

Also, theoretical works by [46], [80], [109], [113], [82], and [26] have been used to study the aspect of optimization. They have been used as well as a basis to develop the optimization model. The model itself is based on five optimization methods: full-factorial design, simple-random sampling, latin-hypercube sampling, hill-climbing, and simulated-annealing.

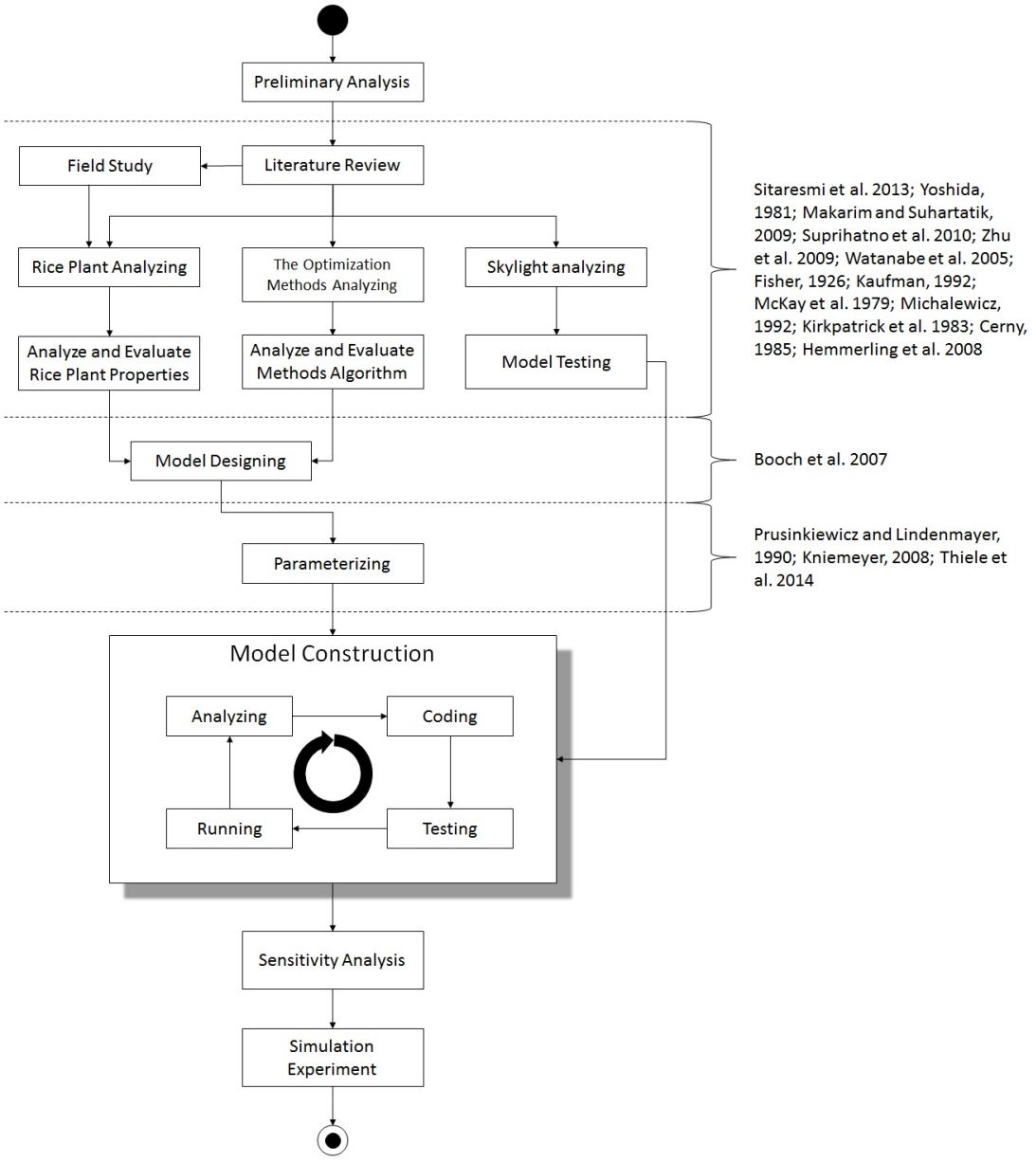

FIGURE 5.1: Research stages

In addition, for the skylight model, the radiation model from [64] has been studied deeply and became the main source for the skylight model; which was merged into the whole model then. The radiation model makes use of a path tracer algorithm [166].

In the next stage, an analysis for each aspect in more detail was conducted. Exclusively with the rice plant, numerous attributes that relate to quantitative parameters and properties have been analyzed and evaluated. For fulfilling the purpose, especially for empirical data about four rice varieties as an object of this research, field studies have been used which had been conducted by several Indonesian centers for rice research (ICRR), mainly from [146] in ICRR Cilamaya city, West Java province, Indonesia.

Regarding the optimization aspect, the algorithms of five optimization methods have been analyzed and evaluated intensely here. Global variables and parameters, and detailed algorithms have been results of this stage. After that, we could move to the next stage (parameterizing stage).

For the skylight aspect, model testing is a task that must be done here. By changing a number of parameters, the skylight model [64] is embedded into the whole model. This stage is already part of model construction.

The concept of object-oriented design [17] is used for model design (in the stage of designing). It is a starting point to code the model in the stage model construction by using XL programming under the modelling platform GroIMP [93] and based on the FSPM concept [94]. And then, in the stage of parameterizing, parameters' identification and calibration [157] based on L-systems and growth grammars is initially done.

In the stage model construction, four cyclic activities are done. By embedding the skylight model, the model is constructed. The model itself consists of three sub-models: rice plant, optimization, and skylight. In this stage, the definition and parameterizing of selected parameters of the rice plant structure is one important thing to do, as they will become independent variables of the objective function of the optimization model.

5.2 Empirical Data

Four Indonesian varieties of rice (*Oryzasativa* L.) are used in this research. The varieties are: Dodokan, Fatmawati, Inpari 9, and Ciherang. They are the popular varieties in Indonesia, and four of some varieties that have a good performance in yield production [98].

Regarding the empirical data of rice morphology that are used in the model, mostly they come from [146]. Their research primarily focused on morphology of Indonesian varieties of rice. Their research was conducted in the experimental garden, ICRR Cilamaya city, West Java province, Indonesia; from September 2011 - January 2012. The research place's geographical coordinates are 6^o 14' 51" South, 107^o 34' 5" East. Several observed morphological aspects include color of leaf and stem, dimensions and development of leaves, stem, panicle, tillers, etc.

5.3 Model Development

Model development here means constructing the whole model from designing stage until implementation. The object-oriented programming paradigm [17] is used to design the model. The class diagram is a main tool used to describe the model. In model construction, XL programming on the model platform GroIMP is used. Based on the FSPM approach, the technical activities of analyzing, coding, testing, and simulating are done cyclically and sequentially. Actually, coding means how to convert the human language (also design result) to specified code that can be understood well by the machine (computer), testing means the process of model procedures or functions assessment to make sure the model runs well based on the purposes, and simulating means the process of model execution to get the result. Revising and improving can be done based on testing results, they possibly let the four activities in model construction be executed cyclically.

The model can produce several results, amongst them a 3-d view and numerical data. A statistical approach is used to process the data to be delivered into a more meaningful and valuable format.

5.4 Sensitivity Analysis and Simulation Experiment

Practically, the sensitivity analysis aims to see the effect strength of the alterations of 15 selected parameters to the value of light interception and to enhance the understanding of the connections between the selected parameters and the objective function that have been defined before. The optimization model "full factorial design" is used in this activity which is implemented to test the parameters of the rice plant variety Dodokan. This analysis is also implemented to test 6 additional parameters, representing characteristic parameters of four Indonesian rice varieties that will be used in a second optimization process where we will interpolate between the four varieties (see Figure 5.2).

In a preliminary study, by using the model we constructed, we successfully proposed several new optimal shapes of the rice plant based on the calculation of intercepted light by the leaves on the main stem. We conducted the simulation experiment through changing only two parameters (parameters starting day for a leaf to bend and leaf angle) of two varieties [163].

In our central optimization study, the purpose of the simulation experiment is to propose a new shape of the rice plant, based on a larger number of shape-describing parameters. As mentioned before, this research used four varieties of Indonesian rice. To propose a new shape of the rice plant, we conducted two simulation experiments by respectively

modifying 15 selected and 6 additional parameters. The 15 selected parameters represent the morphological aspects describing the rice plant in general. For instance the parameter "stem length coefficient" serves as a multiplier of the stem length. The model can propose the optimal coefficient. The optimal coefficient will probably be different for each variety. Thus, it can be seen that the result of the first parameter-modification process is a new optimal shape for each variety (four types of shape based on each variety's characteristics; such as leaf length, leaf width, stem length, etc.).

For getting a new variety shape, the best previous optimal parameter combination (selected one from four types) needs to be tested by using several interpolated characteristic parameters (in this case 6 parameters) based on all four varieties. For example, the multiplier coefficient (the parameter "stem length coefficient" above) is used to be multiplied with the interpolated parameter "stem length". So, from this second optimization process, which involves indeed an interpolation between the existing varieties in the parameter space, a new variety shape is obtained. The virtual experiment itself was conducted in the Laboratory of the Department Ecoinformatics, Biometrics and Forest Growth, Georg-August University of Göttingen, Germany by using a machine with a processor Intel Core2 Quad Q9450 2.66GHz during around 2 months. All results of light interception calculation in the experiment are delivered as relative values with respect to the maximal obtained value.

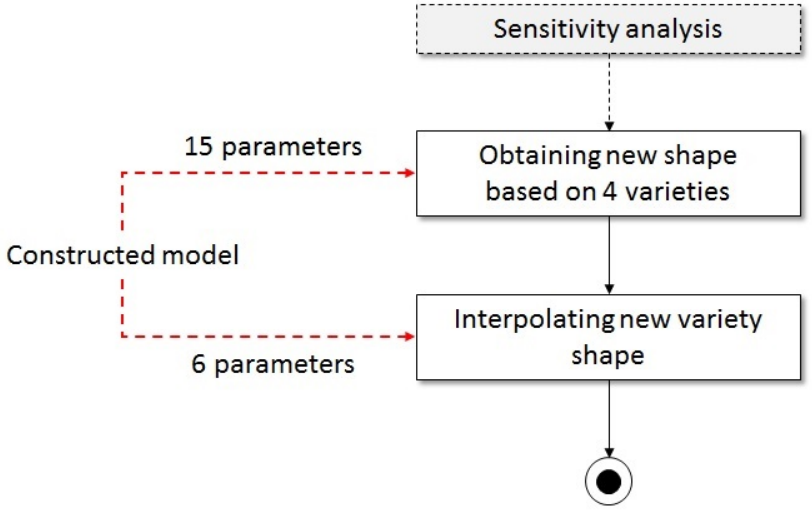

FIGURE 5.2: Two simulation experiments using modification of and interpolation between existing morphologies

Regarding the relative value that we used in our virtual experiments (especially in calculating the light interception value), it is used due to several reasons. The first reason, one of the research objectives is to describe the requirements that make the rice plant

have an optimal shape (in this case, it is related to light interception). Here, the possibilities of the rice plant shape are compared to each other to find the highest value of light interception. For the purpose of comparison, the relative values are sufficient. The second reason is about the availability of empirical data. The main empirical data that we used in the research do not have information directly relating to light interception, they are only related to the general structural parameters of the rice plant. Particularly, we did not measure the absolute values of light-flux density or intercepted radiation energy in the field. So, the relative value use in the light interception calculation is logically appropriate.

Part III

Constructed Model and Result

Chapter 6

Constructed Model

As a coarse level, the constructed model consists of three big clusters (Figure 6.1). The three big clusters are rice plant, skylight, and optimization models. The model rice plant can simulate the development and growth of a single above-ground rice plant of four types of Indonesian rice variety during the vegetative phase. The model rice plant consists of four interconnected classes: $RicePlant$, $Leaf$, $Stem$, and $Internode$. The cardinality (1..* and 1) between the clusters rice plant and skylight means that the skylight model principally can correspond with one or more than one rice plant (i.e., with at least one rice plant).

In addition, the model skylight simulates the sun movement and skylight based on the geographical location of Indonesia. Naturally, the skylight has two types of light: direct and diffuse light. They are represented by two classes Sun and Sky that are parts of the class $SkyLight$.

And the last cluster, the optimization models can find and propose the optimal interpolated varieties based on the calculation of the light interception value. This model consists of five types of optimization method: $FullFactorialDesign$, $SimpleRandomSampling$, $LatinHypercubeSampling$, $HillClimbing$, and $SimulatedAnnealing$. This model itself aims to find the optimal value of the objective function which is to maximize the value of light intercepted by one rice plant by changing the selected parameters of the rice plant structure of four Indonesian rice varieties. The cardinality (1..* and 1..*) between the clusters rice plant and optimization model indicates that the constructed model can simulate one or more than one rice plant (as mentioned before) and apply one or more than one type of optimization. By using two levels of the interpolation process, the optimal interpolated variety is proposed (see chapter research methodology in Figure 5.2).

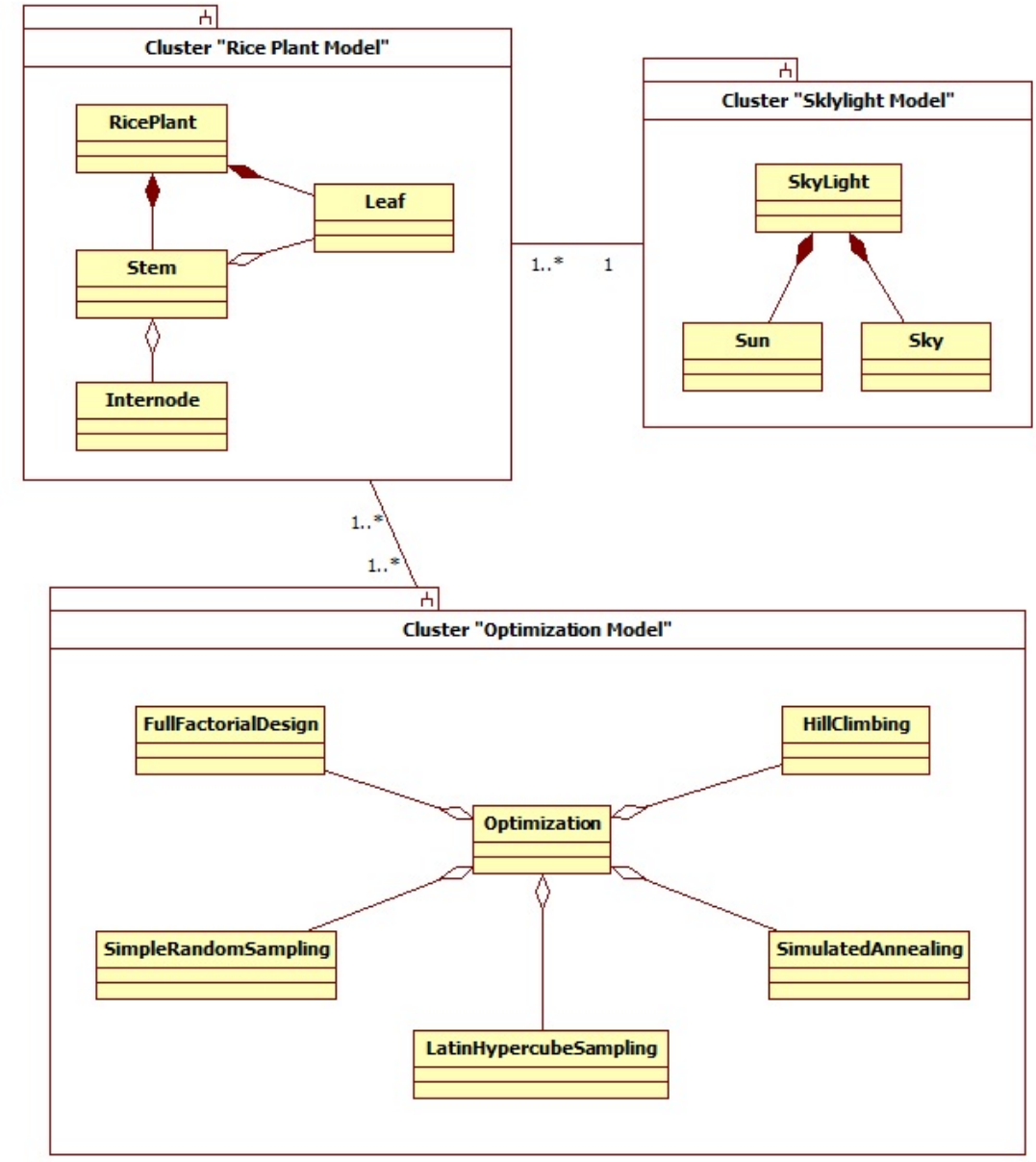

FIGURE 6.1: Class diagram of the constructed model

6.1 Rice Plant Model

The rice plant model is constructed to see the development and growth process of a rice plant in detail. Several model assumptions are used. The main rice plant model assumptions are listed here:

1. The modelled rice plant is a single above-ground rice plant. Its simulated development and growth is restricted to the vegetative phase, where the vegetative organs

involved are stem / tillers and leaves. It does not involve the organs panicle, grain, and root.

2. The values of simulated maximum age within the vegetative phase for the four varieties Dodokan, Fatmawati, Inpari 9, and Ciherang respectively are 51, 57, 60, and 56 days; they come from empirical data of an average plant age minus 65 days (the reproductive phase lasts 35 days and the ripening phase 30 days) [146] [71].

Structurally, the model is depicted in a class diagram in Figure 6.2. The class diagram consists of four classes, they are *RicePlant*, *Leaf*, *Stem*, and *Internode*. As mentioned in the model assumptions, the rice plant is a single above-ground plant and simulated without panicle, grain, and root. All calculations of parameters here are conducted during the vegetative phase.

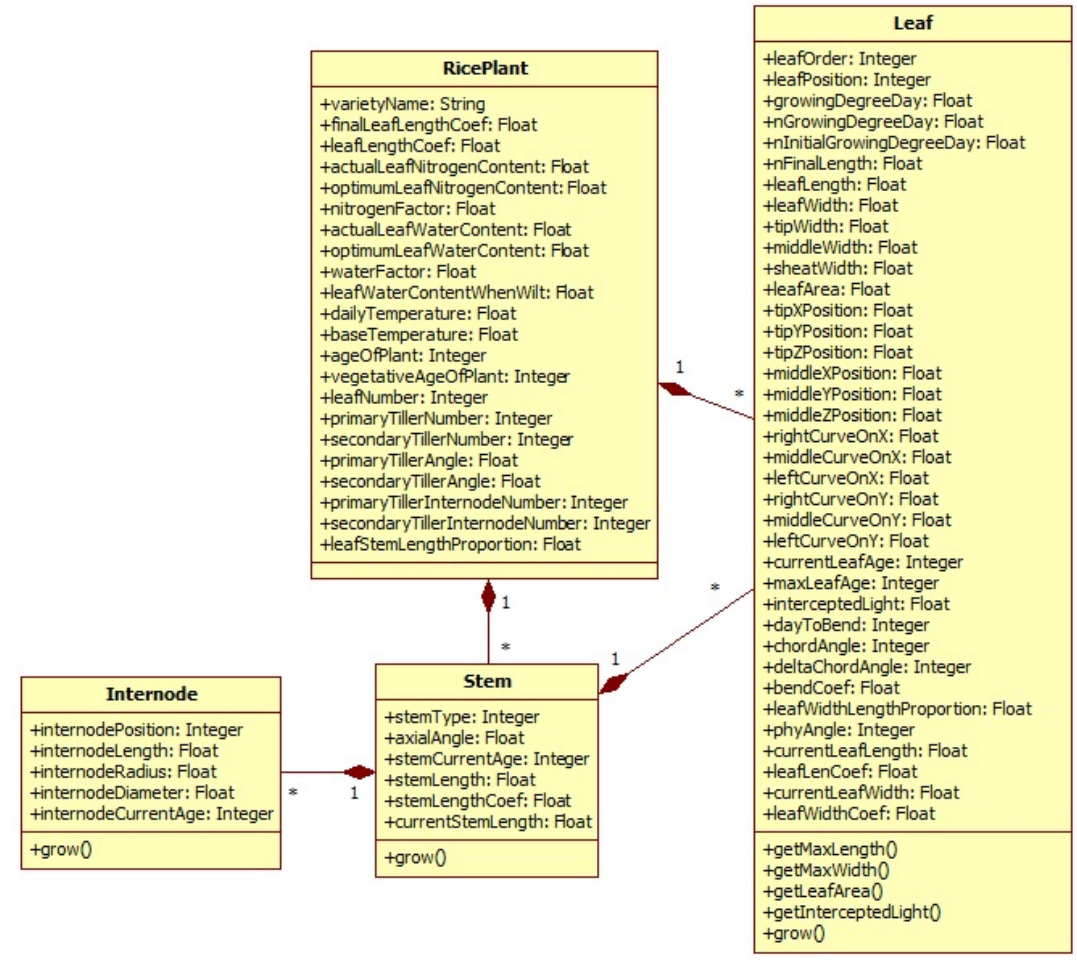

FIGURE 6.2: Class diagram of rice plant model

The parameter *varietyName* in the class *RicePlant* stands for one of four Indonesian rice varieties that are used as research objects in this research. The four varieties are

Dodokan, Fatmawati, Inpari 9, and Ciherang. The parameter $leafOrder$ in the class $Leaf$ is valued by 0, 1, or 2. They indicate that the leaf is located on the main stem, primary tillers, or secondary tillers respectively. The parameter $stemType$ in the class $Stem$ also should be 0, 1, or 2; where it respectively indicates the stem types main, primary, or secondary tiller. In addition, the parameter $internodePosition$ in the class $Internode$ describes the position of the internode in one stem, counted from the bottom. The maximum value of parameter $internodePosition$ is $leafNumber - 1$.

The cardinality in the diagram (Figure 6.2) shows the membership number between two parts (classes). For example, the cardinality between classes $Stem$ and $Leaf$ is 1..*. It means, one and only one stem is possibly able to have many leaves, where the class $Leaf$ is a part of the class $Stem$. A similar cardinality pattern is also used to describe other relationships, such as between classes $RicePlant$ and $Leaf$, $RicePlant$ and $Stem$, and $Stem$ and $Internode$.

Practically, rice leaves grow alternately on the stem with one leaf on one node of the stem with different duration of growth. Here, it is assumed that the first leaf needs around 5 days and the others need 8 days to fully grow. The XL code in Listing 6.1 describes a rice leaf development, especially on the main stem. All commands in the object $Stem$ are done under the condition "$(st == 0)$ and $(lpos < ln + 2)$"; where the parameters st is the stem type, $lpos$ is the position of the last generated leaf on this stem, and ln is the final number of leaves. A stem is categorized into three types; main, primary and secondary stems with parameter st defined as 0, 1, and 2 respectively. The parameter sca is the current age of the stem. It represents the current number of passed days of growth. It is very useful to control at which day the leaf will start to grow on the stem.

The condition "$((lpos == 1)$ and $(sca == 1))$" means that in the first position and first day, the internode and leaf start to grow by calling the modules $Internode()$ and $Leaf()$. The condition "$(lpos < ln + 1)$" is used to make sure that the current position is still smaller than $leafnumber + 1$. For the next internode and leaf, the second condition "$(sca == (5 + ((lpos - 2) * 8)))$" is used. It means they will start to grow at the $5th$ day, $13th$ day, $21st$ day, etc. In addition, the parameter ra is a rotation angle, it represents the phyllotactic angle of the leaf.

```
Stem(st, lpos, sca, ra, and other stem parameters...),
((st == 0)&&(lpos<ln+2)) ==>
   {
      sca++;
   }
   // for the first day
   if ((lpos == 1)  && (sca == 1))
   (
      // calling the module of internode
      Internode(lpos and other Internode parameters...)
```

```
      if (lpos < ln+1)
      (
         [
            // calling the module of leaf
            Leaf(Leaf parameters...)
         ]
      )
      {
         lpos++;
      }
   )
   // for the 5th, 13th, 21st, ..., nth day
   else if (sca == (5+((lpos-2)*8)))
   (
      RH(ra)
      // calling the module of internode
      Internode(lpos and other internode parameters...)

      if (lpos < ln+1)
      (
         [
            // calling the module of leaf
            Leaf(Leaf parameters...)
         ]
      )
      {
         lpos++;
      }
   )

   // calling the module of stem
   Stem(st, lpos, sca, ra, and other stem parameters...);
```

LISTING 6.1: XL code for modelling the stem and leaf development

6.1.1 Rice Leaf Model

6.1.1.1 Model for Calculating Leaf Parameters

[189] developed a leaf shape model of the rice plant to characterize the changing pattern of leaf growth during plant development and to model the changes in morphology of the different leaves as well. Basically, leaves are the main photosynthetic organs that have the most impact on light energy utilization. This is one reason why the model is very fruitful for designing optimal plant shape and visualizing plant growth. We took this model as a basis for our rice leaf model.

The model was developed by [189] based on four experiments carried out in 2005 at the field station of Nanjing Agricultural University, China. In the first experiment, the effect of four nitrogen levels was investigated (0, 82, 165 and 247 $kg\ N\ ha^{-1}$), by using urea as N-fertilizer. In the second experiment, four water regimes were established (70%, 80%, 90% and 100% of soil capacity). These water regimes were applied in four kinds of cultivar of rice (the third experiment): Wuxiangjing 14, Nippobare, Nanjing 16 and Yangdao 6. Finally, treatments consisting of factorial combinations of nitrogen rates and water regimes were applied in the fourth experiment.

The model has several components. The first one is the expansion process of a single leaf on the main stem. It is based on a slow-rapid-slow pattern and formulated in equation (6.1). There, GDD (^oCd) is the number of growing degree days since leaf emergence, it is determined in equation (6.2); $LL_n(GDD)$ (cm) is the length of leaf n on the main stem at GDD; LL_n (cm) is the final length of fully expanded leaf n, it is described in equation (6.3); La and Lb are coefficients (see Table 6.1 for the values); $IGDD_n$ (^oCd) is the initial number of GDD when leaf n begins to grow, it is technically formulated in equation (6.4); GDD_n (^oCd) is the number of GDD required for completing normal growth of leaf n, it is derived from equation (6.5); and FN and FW are factors accounting for nitrogen and water effects, they are described in equations (6.6) and (6.7).

$$LL_n(GDD) = \frac{LL_n}{1 + La \times e^{-Lb \times (GDD - IGDD_n)/\Delta GDD_n} \times \min(FN, FW)} \tag{6.1}$$

$$GDD = \sum (Ti - Tb) \tag{6.2}$$

$$LL_n = \begin{cases} (a_1 \times n^2 + b_1 \times n + c_1) \times FLCV + LCV, \ 1 \leq n \leq (LN - N + 1) \\ (a_2 \times n^2 + b_2 \times n + c_2) \times FLCV + LCV, \ (LN - N + 1) \leq n \leq LN \end{cases} \tag{6.3}$$

$$IGDD_n = \left(\frac{n}{a}\right)^{1/b} - kl \times \Delta GDD_n \tag{6.4}$$

$$\Delta GDD_n = \left(\frac{n + 1.5}{a}\right)^{1/b} - \left(\frac{n}{a}\right)^{1/b} \tag{6.5}$$

$$FN = \begin{cases} 1 - Nla \times (ANCL - ONCL)^2, \ 0 < ANCL < ONCL \\ 1, \ ANCL \geq ONCL \end{cases} \tag{6.6}$$

$$FW = \begin{cases} 1 - Wla \times \left(\frac{AWCL - LW_{wp}}{OWCL - LW_{wp}}\right)^2, \ 0 < AWCL < OWCL \\ 1, \ AWCL \geq OWCL \end{cases} \tag{6.7}$$

In equation (6.2), Ti is the average daily temperature (^oC), and Tb is the base temperature (^oC) for growth. For the case of Indonesian conditions, the daily temperature used is based on the average daily temperature of West Java province (24.6^oC; BPS, 2015); and 12^oC is used as the base temperature [189]. The variable kl in equation (6.4) is

a coefficient that differs with variety (see Table 6.1 for the value of kl, a, and b); the same holds for a_1, b_1, c_1, a_2, b_2, and c_2 in equation (6.3), they are coefficients, where the values of a_1 and a_2 for variety Dodokan, Fatmawati, Inpari 9, and Ciherang respectively are 0.04 and -0.0296, 0.04 and -0.0296, 0.09 and 0.029, and 0.04 and -0.0296 (for other values of coefficients see Table 6.1); $FLCV$ is the change coefficient of final leaf length, where for variety Dodokan, Fatmawati, Inpari 9, and Ciherang respectively the values are 78.09, 109.33, 55.00, and 99.19; and LCV is the change coefficient of leaf length (it is assumed 0); LN is the final leaf number, where for variety Dodokan, Fatmawati, Inpari 9, and Ciherang respectively the values are 14, 15, 17, and 16; and N is the number of elongated internodes on the main stem (here it is 4). In addition, Nla in equation (6.6) is an equation coefficient (see Table 6.1); $ANCL$ is the actual leaf-nitrogen content (see Table 6.1); and $ONCL$ is the optimum leaf-nitrogen content for rice (see Table 6.1). Furthermore, based on equation (6.7), Wla is an equation coefficient (see Table 6.1); $AWCL$ is the actual leaf-water content (see Table 6.1); LW_{wp} is the leaf-water content when a leaf begins to wilt (see Table 6.1); and $OWCL$ is the optimum water content of the leaf [189].

Furthermore, the maximum width of a fully expanded leaf on the main stem can be described by equation (6.8); where n is leaf position; LW_n is the maximum width of the fully expanded leaf n; W_a, W_b and W_c are equation coefficients, where coefficient value of W_a and W_c for variety Dodokan, Fatmawati, Inpari 9, and Ciherang respectively are -0.0104 and -0.0541, -0.092 and 0.5941, -0.0081 and -0.1800, and -0.0087 and 0.0141 (for value of coefficient W_b see Table 6.1). All identical parameters for four Indonesian varieties used in model are acquired from parameterizing process based on [146] empirical data.

$$LW_n(GDD) = LW_n = W_a \times n^2 + W_b \times n + W_c \tag{6.8}$$

For calculating leaf area, equations (6.9) upto (6.20) are used. Essentially, the rice leaf is divided into several simple parts (see Figure 6.3). Triangle, trapezium, and rectangle are basic forms to construct the parts of the rice leaf. Triangle height (tH) is assumed as 0.75 maximum leaf length (mLL; see equation (6.9)), thus trapezium heigh (tRH) can be calculated (equation 6.10). To calculate triangle area (tA), equation (6.11) is used, where mLW is maximum leaf width coming from equation (6.8). The height of trapezium 1 ($tRH1$) can be assumed as 0.75 trapezium height (tRH, see equation 6.12). The width of trapezium 1 ($tRW1$) can be calculated by using equation (6.13), where mSW is maximum leaf sheath that is assumed to be 0.1 mLW (equation 6.14).

No.	Coefficient Name	Value	Source
1.	a	0.01617	based on our parameterizing result
2.	b	0.935	[189]
3.	kl	0.01	[189]
4.	b_1	0.0677	[189]
5.	c_1	0.0241	[189]
6.	b_2	0.7592	[189]
7.	c_2	3.8876	[189]
8.	La	8.65	[189]
9.	Lb	6.26	[189]
10.	Nla	0.0016	[189]
11.	$ANCL$	0.0315	[189]
12.	$ONCL$	0.0315	based on our parameterizing result
13.	Wla	0.001	[189]
14.	$AWCL$	0.8	based on our parameterizing result
15.	$OWCL$	0.8	[189]
16.	LW_{wp}	0.3	[189]
17.	W_b	0.2194	[189]

TABLE 6.1: Coefficient values for length and width calculation

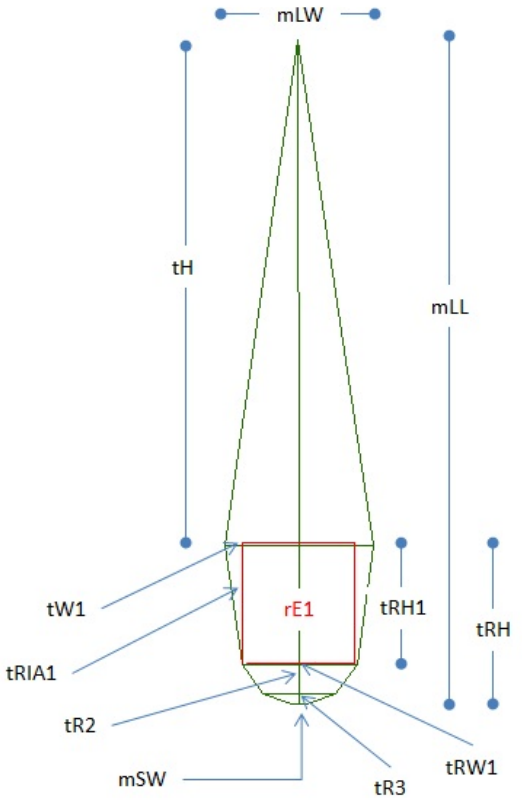

FIGURE 6.3: Schematic view of rice leaf parts; where mLL is maximum leaf length, mLW is maximum leaf width, tH is triangle height, $rE1$ is rectangle 1, tRH is trapezium height, $tRH1$ is trapezium 1 height, $tW1$ is triangle 1 width, $tRW1$ is trapezium 1 width, mSW is maximum leaf sheath width, $tRIA1$ is triangle 1, $tR2$ is trapezium 2, and $tR3$ is trapezium 3.

$$tH = 0.75 \times mLL \tag{6.9}$$

$$tRH = 0.25 \times mLL \tag{6.10}$$

$$tA = \begin{cases} 0.5 \times mLW \times tH \\ 0.375 \times mLW \times mLL \end{cases} \tag{6.11}$$

$$tRH1 = \begin{cases} 0.75 \times tRH \\ 0.1875 \times mLL \end{cases} \tag{6.12}$$

$$tRW1 = 0.75 \times (mLW - mSW) \tag{6.13}$$

$$mSW = 0.1 \times mLW \tag{6.14}$$

In addition, the width of triangle 1 ($tW1$) is calculated by using equation (6.15). The areas of triangle 1 ($tRIA1$) and rectangle 1 ($rEA1$) are calculated by using equation (6.16) and (6.17) respectively. The area of trapezium 1, 2, and 3 ($tRA1$, $tRA2$, and $tRA3$) are calculated by using equation (6.18), (6.19), (6.20) respectively; by assuming that $tRA2$ is 0.25 $tRA1$, and $tRA3$ is 0.25 $tRA2$. Finally, the leaf area (LA) is calculated by using equation (6.21)

$$tW1 = mLW - tRW1 \tag{6.15}$$

$$tRIA1 = 0.5 \times tW1 \times tRH1 \tag{6.16}$$

$$rEA1 = tRH1 \times tRW1 \tag{6.17}$$

$$tRA1 = 2 \times tRIA1 + rEA1 \tag{6.18}$$

$$tRA2 = 0.25 \times tRA1 \tag{6.19}$$

$$tRA3 = 0.25 \times tRA2 \tag{6.20}$$

$$LA = tA + tRA1 + tRA2 + tRA3 \tag{6.21}$$

In XL programming, a single leaf is constructed by using the method $NURBSSurface$ which is instantiated by an *instantiation rule*. $NURBSSurface$ is used to construct the surface of the leaf by using the attribute $BSplineSurface$. The leaf surface itself is defined as an instance of the class $ExtrudedSurface$ by using the method $BSplineOfVertice$ that relates to the profile ($profile$) and the trajectory ($traj$) of a leaf. The parameter $profile$ itself defines the profile of the leaf through 10 parameters, and the parameter $traj$ defines the trajectory of the leaf through 12 parameters. Listing 6.2 describes the definition of a leaf and its parameters in XL code; and the schematic 3-d view of the leaf model is depicted by Figure 6.4.

```
// the module of leaf
const Shader leafShader = shader("Leaf");

module Leaf(
   int leafCode,
   float theta,
   float dayToBend,
   float lightPower,
   int leafPosition,
   int leafCurrentAge, int leafMaxAge,
   float startOnX, float startOnY, float startOnZ,
   float tipOnX, float tipOnY, float tipOnZ,
   float middleOnX, float middleOnY, float middleOnZ,
   float sheathWidth, float middleWidth, float tipWidth,
   float edgeX1, float edgeX2, float edgeY1, float edgeY2,
   float rightCurveOnX, float middleCurveOnX, float leftCurveOnX,
   float rightCurveOnY, float middleCurveOnY, float leftCurveOnY,
   float maximumLength, float maximumWidth, float leafArea,
   int accMaxAge)

// instantiation rule
==>
{
   VertexList profile = new VertexListImpl(new float[]
   {
      edgeX2, edgeY1,
      rightCurveOnX, rightCurveOnY,
      middleCurveOnX, middleCurveOnY,
      leftCurveOnX, leftCurveOnY,
      edgeX1, edgeY2
   }, 2);

   VertexList traj = new VertexListImpl(new float[]
   {
      startOnX, startOnY, startOnZ, sheathWidth,
      middleOnX, middleOnY, middleOnZ, middleWidth,
      tipOnX, tipOnY, tipOnZ, tipWidth
   }, 4);

   ExtrudedSurface surface = new ExtrudedSurface
   (
         new BSplineOfVertices(profile, 3, false, false),
        new BSplineOfVertices(traj, 2, false, false)
```

```
   ).(setUseScale(true));

   NURBSSurface s = new NURBSSurface(surface);
    s.setShader(new RGBAShader(0.2,0.4,0));
   s.setShader(new AlgorithmSwitchShader(leafShader, leafShader));
} s;
```

LISTING 6.2: XL code for defining the leaf and its parameters

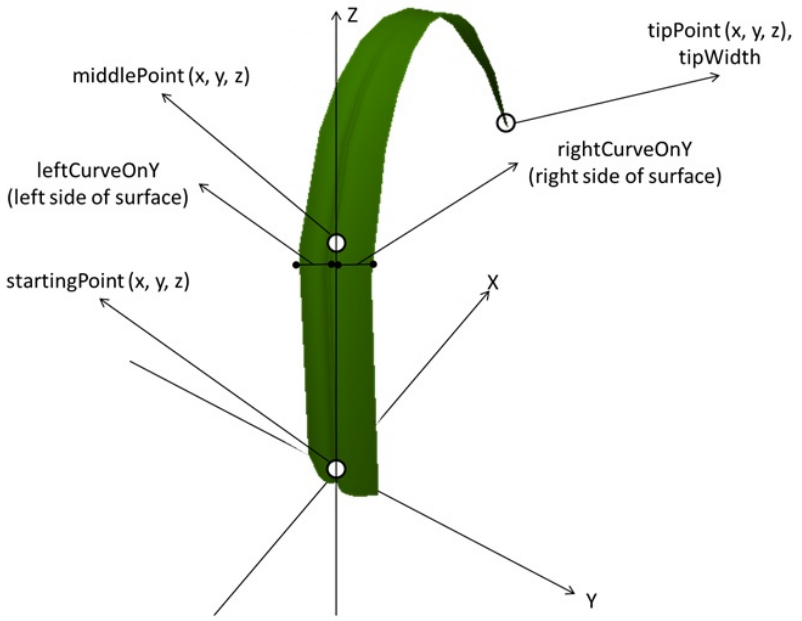

FIGURE 6.4: Schematic 3-d view of single leaf model

The model from [189] is implemented here, especially for calculating the maximal length and width of the rice leaf. The Listing 6.3 is the XL code of the function $getLeafLength()$ to implement the leaf length calculation from equation (6.1) upto (6.7) that returns the value of leaf length for each leaf rank. Also for calculating the width of the leaf for each rank, the model of [189] is implemented (equation (6.8)) in Listing 6.4.

```
float getLeafLength(int n, int accMaxAge)
{
   Parameters definition....

   gDD = accMaxAge * (tI - tB);
   if (aNCL < oNCL)
   {
      fN = 1 - nLa * Math.pow((aNCL - oNCL), 2);
   }
   else
   { fN = 1;}

   if (aWCL >= oWCL)
   {
      fW = 1;
   }
   else
   {
      fW = 1 - wLa * Math.pow((aWCL - lWwp)/(oWCL - lWwp), 2);
```

```
    }
    dGDDn = Math.pow(((n + 1.5)/a), (1/b)) - Math.pow((n/a),
       (1/b));
    iGDDn = Math.pow((n/a), (1/b)) - (kl * dGDDn);
    if ((1 <= n) && (n < (LN-N+1)))
    {
       lLn =  ((a1 * n * n) + (b1 * n) + c1) * fLCV + lCV;
    }
    else
    {
       lLn =  ((a2 * n * n) + (b2 * n) + c2) * fLCV + lCV;
    }
    lLnGDD = inMCoef * ((lLn / (1 + (lA * Math.exp(-lB*(gDD-
       iGDDn)/dGDDn))))*Math.min(fN, fW));
    return(lLnGDD);
};
```

LISTING 6.3: XL code for calculating the length a of leaf

```
float getLeafWidth(int n)
{
    Parameters definition...
    lWn = inMCoef * (wA*n*n + wB*n + wC);
    return(lWn);
}
```

LISTING 6.4: XL code for calculating the width of a leaf

For calculating leaf area, equations (6.9) upto (6.21) are implemented in Listing 6.5. Parameters used are leaf length ($lLnGDD1$) and width ($lWn1$) for each leaf rank.

```
float getLeafArea(float lLnGDD1, float lWn1)
{
    Parameters definition...
    tH = 0.75 * lLnGDD1;
    tA = 0.5 * tH * lWn1;
    tRH = 0.25 * lLnGDD1;
    tRH1 = 0.75 * tRH;
    tRW1 = 0.75 * (lWn1 - lWn1 / 10);
    tW1 = lWn1 - tRW1;
    tRIA1 = 0.5 * tW1 * tRH1;
    rEA1 = tRH1 *tRW1;
    tRA1 = (2 * tRIA1) + rEA1;
    tRA2 = 0.25 * tRA1;
    tRA3 = 0.25 * tRA2;
    leafArea = tA + tRA1 + tRA2 + tRA3;
    return(leafArea);
};
```

LISTING 6.5: XL code for calculating the leaf area

6.1.1.2 Leaf Bending Mechanism

The leaf bending mechanism can be described by Figure 6.5 (for the case where M is above the chord line) and 6.6 (for the case where M is below the chord line), where the

original leaf is illustrated by the green line and the bended leaf is illustrated by the blue line. M itself is the middle point of the bended leaf. We call this mechanism the simple leaf bending mechanism.

Leaf bending depends on three parameters: leaf chord angle (θ), parameter L (sum of two equal chord lengths $\overline{SM}$ and $\overline{MT}$), and bending coefficient ($p = \frac{H}{0.5L} = \frac{2H}{L}$, see Figure 6.5 or 6.6). p will be $-1 < p < 1$, where $p < 0$ is assumed if M (blue color, the middle point of the bended leaf) is below ST (see Figure 6.6). Parameter η is the angle between leaf chord and line of M. It can be measured by using equation (6.22). The parameter line $\overline{ST}$ (blue) is calculated by equation (6.23). The parameter d is the coordinate Z position of point M (blue). For M above the chord line, d is calculated by using equation (6.24); and for M below the line, d is calculated by using equation (6.25). The parameter $tipOnZ$ is used to locate the point of the leaf tip in ordinate Z. It is can be calculated by equation (6.26). And the parameter $tipOnX$ is used to plot the point of the leaf tip in ordinate X. It is calculated by using equation (6.27).

$$\eta = \arcsin(p) \tag{6.22}$$

$$\overline{ST} = 2 \times \sqrt{(0.5L)^2 - H^2} \tag{6.23}$$

$$d = 0.5L \times \sin(\phi + \eta) = 0.5L \times \sin\left(\frac{\pi}{2} - \theta + \arcsin(p)\right) \tag{6.24}$$

$$d = 0.5L \times \sin\left(\frac{\pi}{2} - (\theta + \eta)\right) = 0.5L \times \sin\left(\frac{\pi}{2} - (\theta + \arcsin(p))\right) \tag{6.25}$$

$$tipOnZ = \overline{ST} \times \sin(\phi) = \overline{ST} \times \sin\left(\frac{\pi}{2} - \theta\right) \tag{6.26}$$

$$tipOnX = \overline{ST} \times \cos(\phi) = \overline{ST} \times \cos\left(\frac{\pi}{2} - \theta\right) \tag{6.27}$$

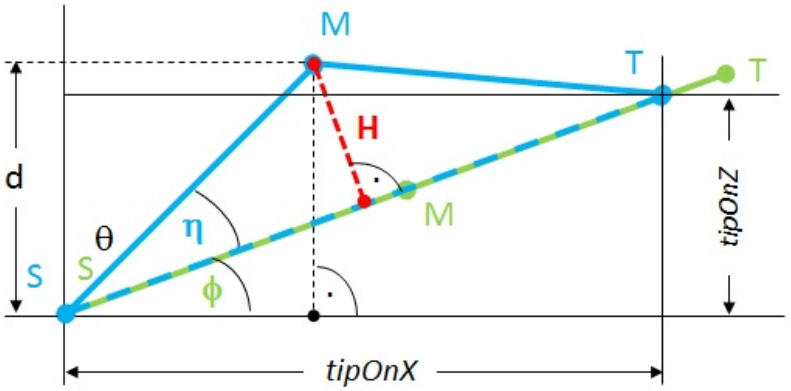

FIGURE 6.5: Schematic view of bending mechanism, when M is above the leaf chord

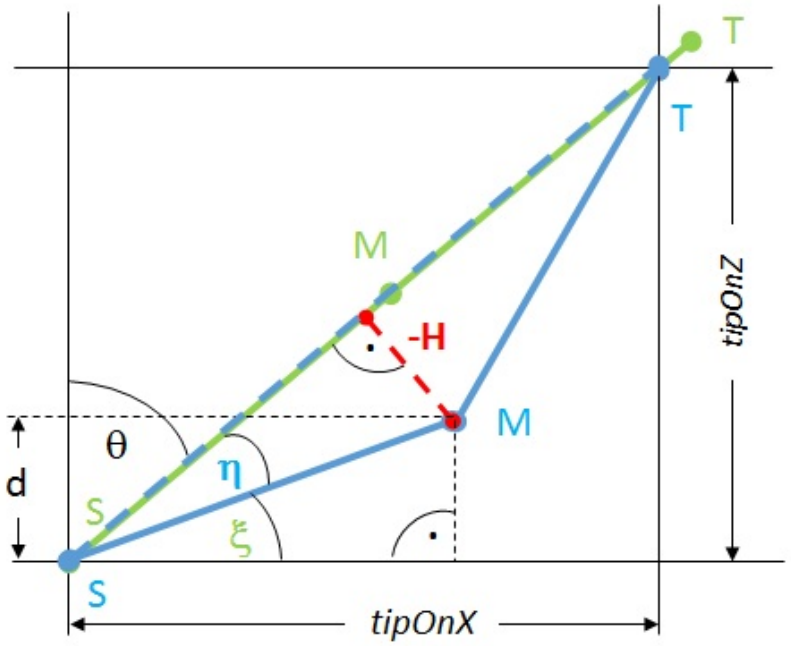

FIGURE 6.6: Schematic view of bending mechanism, when M is below the leaf chord

6.1.2 Stem and Tillers

6.1.2.1 Stem Development

A stem consists of many nodes and internodes. The stem length is the total of all internode lengths in one stem. The highest internode in one stem is the longest internode, and the internode length gradually decreases until the lowest internode [105]. Practically, a short and hard stem is expected, as it is stronger and more balanced, and it is more responsive to nitrogen fertilization [77], [183], and [105].

To model the stem and tillers development, two general model assumptions are used. They are: the model only uses primary and secondary tillers and all tillers are assumed alive during the vegetative phase. General rules are also used in the model, they are:

1. The upper internodes are longer than the lower ones [180].
2. Primary and secondary tillers have around 12 and 10 internodes respectively [189].
3. The growth length of the stem in the vegetative phase is shorter than in the reproductive phase. This reflects that plant height increases more rapidly in the reproductive phase [183].

The stem length can be calculated by using equation (6.28), where parameter $iPos$ is the position of internode (internode rank), a is the internode coefficient, and $maxPos$ is the maximal number of internodes. The position of an internode depends on the age of plant, and it will be different in each variety; where, in average, the first internode consumes 5 days to fully grow, and 8 days for the next internodes. Based on the empirical data [146], the internode coefficients (a) for variety Dodokan, Fatmawati, Inpari 9, and Ciherang were successfully parameterized. They are respectively 0.8286, 0.8551, 0.7261, and 0.7349. And, for each variety, the $stemLength$ at the end of vegetative and reproductive phases respectively is 0.2164 m and 0.7257 m, 0.2843 m and 0.8693 m, 0.2807 m and 0.8085 m, and 0.2849 m and 0.7448 m.

$$stemLength = \sum_{iPos=1}^{maxPos} (iPos)^a \tag{6.28}$$

Basically, stem and internode are defined as modules with several specified parameters. In XL code, both modules are declared like in Listing 6.6. The stem needs to have many parameters to control its development. Parameter $stRad$ is a stem radius, it has a fixed value; $stType$ is a stem type where 0, 1, and 2 represent main stem, primary, and secondary tillers respectively; $leafPosOnSt$ is the rank of the last leaf on stem; $counter$ is a counter number; $stCurAge$ represents the current age of the stem; $axAngle$ symbolizes an axial angle, it is used by tillers.

In addition, the parameter $difCoef$ is a coefficient used to control the height of tillers; $leafNum$ is the number of leaves; $rotAngle$ is a rotation angle, it represents the phyllotaxic angle of leaves; $tilBendNum$ and $angleDec$ are coefficient influencing of tiller bending and angle decrement used to control the bending process of tiller; and $anglePos$ is a position used to control when the tiller bending mechanism is done (see the usage in Listing 6.6).

The parameters of the module Internode are *length* and *radius*, they are used to control length and radius of the internode; *inCurAge*, it is the current age of the internode; *inPos*, it is the rank of the internode in the stem; *intArea*, it is used to record the value of internode area; *stType*, it is the type of stem; and *stLightPower*, it is used to record how much light is captured by the internode.

```
// the module of stem
module Stem(
   float stRad, int stType, int leafPosOnSt,
   int counter, int stCurAge, float axAngle, int difCoef,
   float leafNum, float rotAngle, int tilBendNum,
   float angleDec, float anglePos) extends Sphere(0.0025)
{
   {
      setShader(nodemat);
   }
};

// the module of internode
const Shader stemShader = shader("Stem");
module Internode(
   super.length, super.radius, int inCurAge, int inPos,
   float intArea, int stType, float stLightPower)
   extends Cylinder(length, radius)
{
   {
      setShader(new RGBAShader(1, 0.5, 0.01));
      setShader(new AlgorithmSwitchShader(stemShader,
      stemShader));
   }
};
```

LISTING 6.6: XL code for defining the stem and internode modules

6.1.2.2 Main Stem and Tillers Relationship

For the relationship between the main stem and a tiller, the variations of axial angles used are between 30^o-45^o for primary tillers and 15^o-25^o for secondary tillers, approximately [146]. The schematic figure of the relationship is shown in Figure 6.7; where the angle between line segments 0 and 1 is the maximum angle between main stem and the primary tiller; and the angle between line segments 0 and 2 is the tiller angle after internode elongation. And, the axial angle between a parent module and its daughter tiller increases with the number of tillers produced by the parent axis above and on the same side as the daughter tillers (adopted from [175]).

The relationship between main stem and tillers can be controlled through the module *Stem* with stem type 1 or 2, primary and secondary tillers (Figure 6.7). It can be seen in Listing 6.7.

Principally, the development mechanism of primary and secondary tillers is similar. It is only distinguished by the values of the parameters stem type and axial angle. The

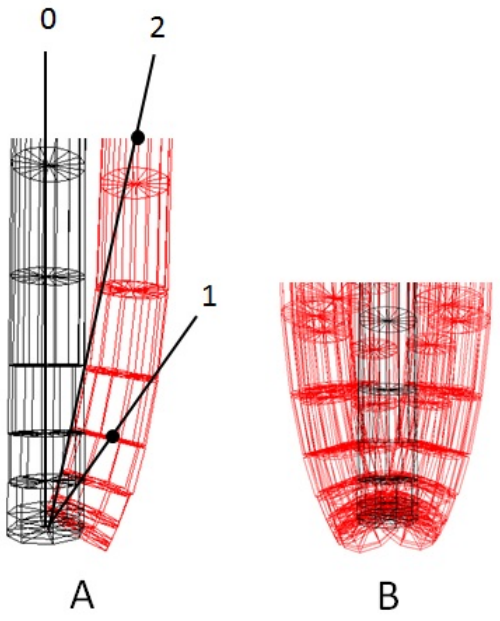

FIGURE 6.7: Schematic 3-d view of stem and tillers relationship; A. Stem and single tiller relationship, B. Stem and several tillers relationship.

parameter dc is a different coefficient, it is used to control the height of tillers compared by the primary tiller. The other way to control the height of tillers is by modifying the parameter ln (leaf number). Parameter a in statement $RU(a)$ is an axial angle, it will be decreased regularly by using the statement $a = -ad$, where ad is an angle decrement. And, the variable ap is used to control the rotation of the stem.

```
Stem(Stem parameters...),
((st == 1)&&((lpos < ln-dc))) ==>
   {
      sca++;
   }

   //for tbn times bending
   if (n < tbn + 1)
   (
      //for first day
      if ((lpos == 1)  && (sca == 1))
      (
         RU(a)

         // calling the module of internode
         Internode(internode parameters...)

         if (lpos < (ln-dc))
         (
            [
               // calling the module of leaf
               Leaf(Leaf parameters...)
            ]
         )

         {
            a=-ad; // for tillers bending
            lpos++;
            n++;
```

```
        }
    )
    //in every  5th, 13th, 21th, ..., nth day
    else if (sca == (5+((lpos-2)*8)))
    (
        // age = 0

        RH(ra)
        {
            ap = ap + ra;
            if(ap>=360)
            {
                ap = ap - 360;
            }
        }

        RH(360-ap)
        RU(a)
        Internode(0, sr, sca, n, 0, st, 0)
        RH(ap)

        if (lpos < (ln+1-dc))
        (
            [
                // calling the module of leaf
                Leaf(Leaf parameters...)
            ]
        )
        {
            a=-ad; // for tillers bending
            lpos++;
            n++;
        }
    )
)
else
(
    //if (sca == (5+((lpos-2)*8)))
    if (sca == (5+((lpos-2)*8)))
    (
        // age = 0
        RH(ra)
        {
            ap = ap + ra;
            if(ap>=360)
            {
                ap = ap - 360;
            }
        }

        RH(360-ap)
        RU(0)
        Internode(0, sr, sca, n, 0, st, 0)
        RH(ap)

        if (lpos < (ln+1-dc))
        (
            [
                // calling the module of leaf
                Leaf(Leaf parameters...)
            ]
        )
        {
            a=-ad; // for tillers bending
            lpos++;
            n++;
        }
    )
)

// calling the module of stem
Stem(Stem parameters...);
```

Listing 6.7: XL code for constructing stem and tiller relationship

6.2 Skylight Model

The skylight model is configured according to the class diagram in Figure 6.8. It utilizes a Monte-Carlo pathtracer which is a part of the software GroIMP [86] and its configurable sky model which was successfully designed by Hemmerling and Buck-Sorlin (unpubl.) based on concepts from [55] and [64].

Principally, skylight is made up of two light types: direct and diffuse lights that are produced by sun and sky respectively. In the object oriented concept, they are represented by the classes *Skylight*, *Sun*, and *Sky*; where the classes *Sun* and *Sky* are aggregated in the class *Skylight*. Both aggregated classes are described by seven global parameters: $numberOfRays$, $numberOfRaysBounce$, *distance*, *latitude*, *time*, $dayOfYear$, and $powerDensity$. The parameter $numberOfRays$ is the assumed number of rays emitted by the light source. The parameter $numberOfRaysBounce$ is the value describing the maximal number of reflections of a ray (recursion depth). In this model, the values 1000000 and 3 are used to define the parameters $numberOfRays$ and $numberOfRaysBounce$.

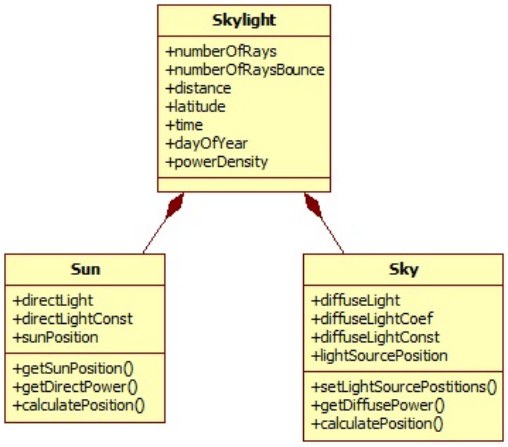

FIGURE 6.8: Class diagram of skylight model

In addition, the parameter *distance* is used to initiate the position of the light source relative to the plant. It is initiated as 100. For the geographical position of the place, it is parameterized by *latitude*. The value of simulated latitude is -7, that means 7^o South Latitude. It represents the geographical location of the province West Java, Indonesia ($5^o50'$ - $7^o50'$ South Latitude). Indeed, the earth locations in between that positions have a similar sun movement and skylight condition. The sun shines for 13 hours on average during a year. The sun movement per hour is visualized in Figure 6.9.

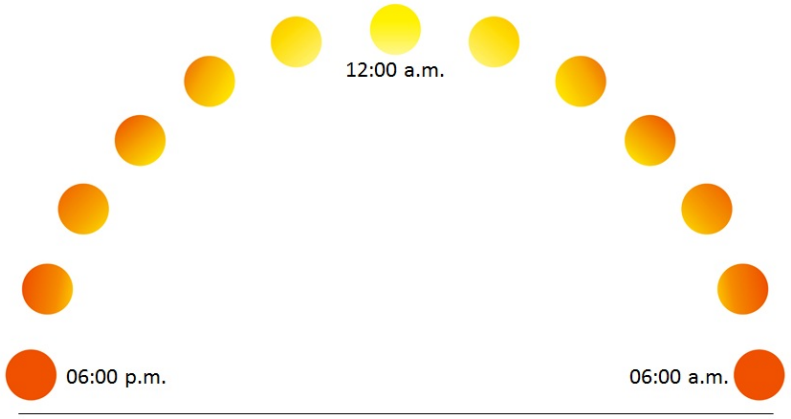

FIGURE 6.9: Sun movement per hour in geographical positions of Indonesia

And two further parameters *time* and $dayOfYear$ are used to define the initiated hour in a day and the initiated day number in a year. In this model, the values 0 and 0 are used to initially define these global parameters respectively. Value 1 (the value coming after the first) for both parameters represents the first hour of 24 hours in a day and the first of January in a year, respectively. The parameter *powerDensity* is an accumulated value of direct and diffuse light. They come from the classes *Sun* and *Sky* respectively that relate to their position, where the position strongly depends on the parameters *distance*, *latitude*, *time*, and $dayOfYear$.

```
module DiffSingLight (float pow) extends LightNode()
{
   {setLight(new DirectionalLight().(setPowerDensity(pow)));}
}

module Sky(float t, int da, float pow) ==>
   RL(90) [ for (int i = 1; i<=12; i++) ([ RU(i*360/12) RL(-11.7)
      M(dist) RL(180) DiffSingLight(i*pow*0.003218)] )]
   [ for (int i = 1; i<=12; i++) ([ RU(20) RU(i*360/12)
      RL(-34.2) M(dist) RL(180) DiffSingLight(i*pow*0.01163)] )]
   [ for (int i = 1; i<=12; i++) ([ RU(40) RU(i*360/12)
      RL(-54.9) M(dist) RL(180) DiffSingLight(i*pow*0.019812)] )]
   [ for (int i = 1; i<=12; i++) ([ RU(60) RU(i*360/12)
      RL(-71.1) M(dist) RL(180) DiffSingLight(i*pow*0.023022)] )]
   [ for (int i = 1; i<=12; i++) ([ RU(80) RU(i*360/12)
      RL(-82.8) M(dist) RL(180) DiffSingLight(i*pow*0.018522)] )]
   [ for (int i = 1; i<=12; i++) ([ RU(80) RU(i*360/12)
      RL(-89.1) M(dist) RL(180) DiffSingLight(i*pow*0.007096)])];

module Sun(int t, float day, float pow) extends LightNode()
{
   {setLight(new DirectionalLight().(setPowerDensity(pow)));}
} ==>  TextLabel(t-1).(setColor(new Color3f(0,0,0)));

LightModel lm = new LightModel(1000000, 3);
```

LISTING 6.8: XL code for defining the sky and sun modules

We now approximate a virtual sky that can produce diffuse light, 72 different positions of light sources are initiated to model the virtual sky dome. The parameter *powerDensity* for the class *Sky* depends on $diffuseLightCoef$ and $diffuseLightConst$. The parameter $diffuseLightCoef$ is defined based on each 6 groups of light source position. The $diffuseLightCoef$ values of group are: 0.003218, 0.01163, 0.019812, 0.023022, 0.018522, and 0.007096. And the parameter $diffuseLightConst$ is a constant factor for diffuse light. It is defined as the constant value 0.45, whereas the parameter *directLightConst* in the class *Sun* is 1 - $diffuseLightConst$.

Two important modules of the skylight model are the modules *Sky* and *Sun*. The virtual sky is constructed by placing 72 light sources in different places, where each light source is configured by the method *LightNode*() by using the module $DiffSingLight$() (see Listing 6.8) that refers to the method *DirectionalLight*(). The parameters used here are *t* as time, *da* as day, and *pow* as a density power. The virtual sky itself is illustrated in Figure 6.10.

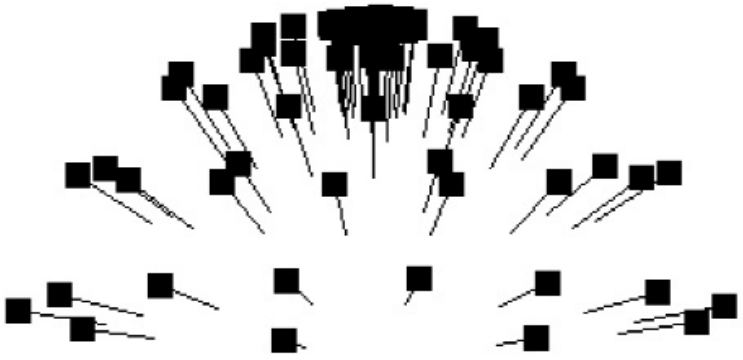

FIGURE 6.10: Schematic view of virtual sky dome

On the other hand, the module *Sun* directly extends to the method *LightNode*(), where light is set by the method *DirectionalLight*(). The module *Sun* uses the same parameters as the module *Sky*. The light model itself is defined by *lm*; where the last command line of Listing 6.8 is used to add a light model to the scene by identifying the number of random rays and reflections. Regarding the density power (*pow*), it totally depends on the position of the light source itself; where the position is produced by the function *calculatingPosition*() (Listing 6.9). The function, for returning a value of density power (*pow*), needs three kinds of parameter: *latitude*, it represents the geographical position of one place on the earth; *day*, it represents the exact day in one year; and *time*, it represents the time in one day. The parameter density power (*pow*) for both *Sun* and

Sky can be obtained by calling the *Sun* and *Sky* procedures in Listing 6.9.

```
Sun((t==24?1:t+1),(t==24?d+0:d),(1 - diffLight)*calcPos
   (latitude, dayOfYear, timeForLight)[3]).(setTransform
   (new Matrix4d(mat, distfact*v, 1)));

Sky ==>
   hm:Sky(dayOfYear, timeForLight,diffLight*calcPos
   (latitude, dayOfYear, timeForLight)[3]);

public float[] calculatingPosition (float latitude, float day,
   int time)
{
   Parameter definition....
   return float result[] = {x,y,z,S,S0Dir*3600*1367*0.6,S0Diff*
   3600*1367*0.6,(S0Dir+S0Diff)*3600*1367*0.6,S0*3600*1367};
}
```

LISTING 6.9: XL code for calculating the position

6.3 Choice of Parameters for Optimization

In this research, 15 selected parameters of rice plant structure are used to optimize the value of light interception; even though the model can adopt more than 15 optimizing parameters. The explanation of the 15 selected parameters is delivered here:

1. Parameter *dayToBend*. It is the "starting day to bend", the day when one leaf starts to bend. Its range is between 1 and 8, with the incremental value 1.0 [105].

2. Parameter *basicChordAngle*. It is the basic leaf chord angle. This parameter is strongly interconnected with parameter number 14 (*deltaChordAngle*). Its range is between 28 and 92 with the incremental value 1.0 (in ^{o}). The value range of the leaf chord angle actually is between 90 and 120 (*basicChordAngle* $\pm$ *deltaChordAngle*) and can be used to describe also the 'recurved' leaf position. A recurved leaf position is a leaf position where the leaf has a chord angle of more than 90^o [69]. Schematically, it is illustrated by Figures 6.11 upto 6.14.

3. Parameter *stDia*. This is the stem diameter. Its range is between 0.0022 and 0.0045 with the incremental value 0.0001 (in m) [146].

4. Parameter p $(2H/L)$. It is a coefficient of leaf bending, where the range is between -0.3 and 1.0 with the incremental value 0.01. Schematically, it is illustrated by Figure 6.12.

5. Parameter *priTilAngle*. It is the angle of primary tillers. Its range is between 30 and 45 (in ^{o}) with the incremental value 1.0. The range represents the minimum and maximum value of the observed tiller angle [146].

6. Parameter $secTilAngle$. It is the angle of secondary tillers. Its range is between 15 and 25 (in ^{o}) with the incremental value 1.0 [146].

7. Parameter $priTilNum$. It is the number of primary tillers that has a range between 3 and 12 with incremental value 1.0. The range is adopted based on data of productive tillers [155], and with the assumption that the proportion between primary and secondary tillers is 1 : 2 [183][105].

8. Parameter $secTilNum$. It is the number of secondary tillers. Its range is between 5 and 23 with incremental value 1.0 [155][183][105].

9. Parameter $prTiIntNum$. It is the number of primary tiller internodes. Its range is between 1 and 12 internodes with incremental value 1.0 [189], assuming that their internode coefficient value is the same.

10. Parameter $seTiIntNum$. It is the number of secondary tiller internodes. Its range is between 1 and 10 internodes with incremental value 1.0 [189].

11. Parameter $stLenCoef$. It is a coefficient of stem length, relating actual stem length to an assumed average (or normal) stem length. Based on data of Indonesian rice plants [155], respectively the shortest and longest heights of rice plant are 0,7 m and 1,4 m. By using the plant height – stem proportion value 0.76 (on average; [146], the shortest and longest stems of a rice plant can be calculated, they are respectively 0,53 m and 1,07 m. Based on empirical data, the average of rice plant stem length is 0.79 m. According to the explanation, the range of the coefficient of stem length can be defined as 0.68 – 1.36 with an incremental value 0.01.

12. Parameter $leStLenProp$. It is the proportion value between leaf length and stem length. Based on empirical data, the shortest and longest rice plant leaf are respectively 0.48 and 0.59 m [146]. The leaf – stem length proportion ($leStLenProp$) is calculated then by dividing the leaf length with the average of stem length, where its range can be obtained as lying between 0.60 and 0.75 (with incremental value 0.01). From this proportion value, the current leaf length (equation (6.29)) and leaf length coefficient (equation (6.30)) of each variety can be calculated.

$$curLeafLen = curStemLen \times leStLenPro \tag{6.29}$$

$$leafLenCoef = curLeafLen/leafLen \tag{6.30}$$

13. Parameter $leafWidLenProp$. It is the proportion of leaf width to length. Based on empirical data, the shortest and longest width of a rice plant leaf are respectively

0.011 and 0.019 m [146]. According to the data, the proportion of leaf width to length ($leafWidLenProp$) is calculated then by dividing the leaf width with the average of leaf length, where its range can be obtained as lying between 0.022 and 0.038 (with incremental value 0.001). From this proportion value, the current leaf width (equation (6.31)) and leaf width coefficient (equation (6.32)) can be calculated.

$$curLeafWid = curLeafLen \times leafWidLenProp \tag{6.31}$$

$$leafWidCoef = curLeafWid/leafWid \tag{6.32}$$

14. Parameter $deltaCA$ ($deltaChordAngle$). Based on the average value of leaf number, and interconnected with parameter 2 ($basicChordAngle$), the range of $deltaCA$ is -2.00 – 2.00 (in ^{o}, with incremental value 0.01).

15. Parameter $varPhi$ (see Figure 6.15). The phyllotactic pattern of the rice plant is the distichous pattern [72]; i.e., a two-ranked phyllotaxy (distichous pattern) with rows of leaves that may not lie exactly, but nearly 180^o apart [13]. Its range is assumed 90 – 270 (in ^{o}, with incremental value 1).

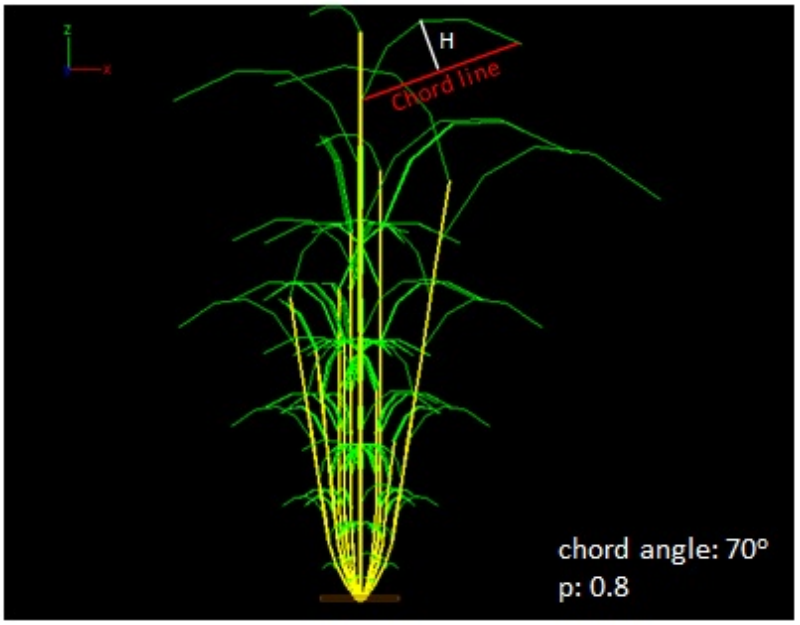

FIGURE 6.11: Schematic view of leaf chord angle (70^o) and bending coefficient (p=0.8)

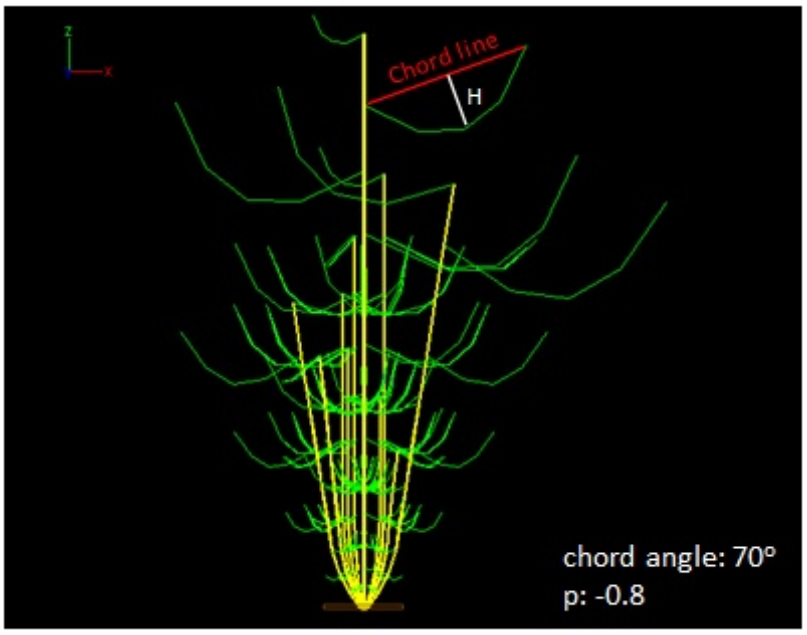

FIGURE 6.12: Schematic view of leaf chord angle (70^o) and bending coefficient (p=-0.8)

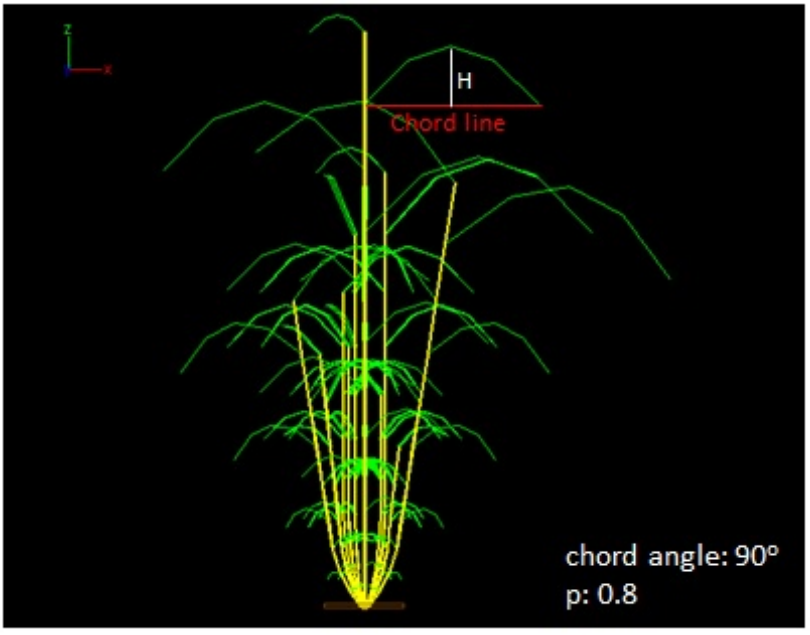

FIGURE 6.13: Schematic view of leaf chord angle (90^o) and bending coefficient (p=0.8)

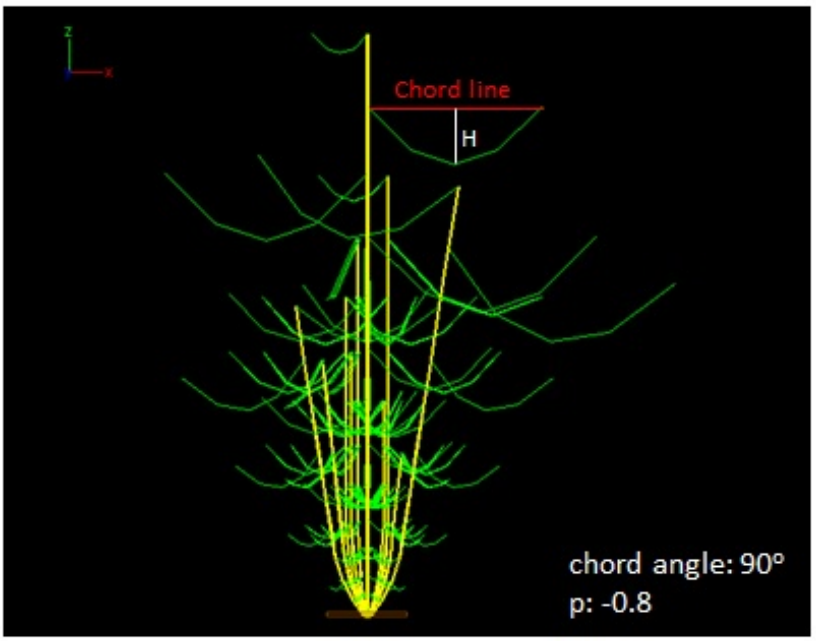

FIGURE 6.14: Schematic view of leaf chord angle (90^o) and bending coefficient (p=-0.8)

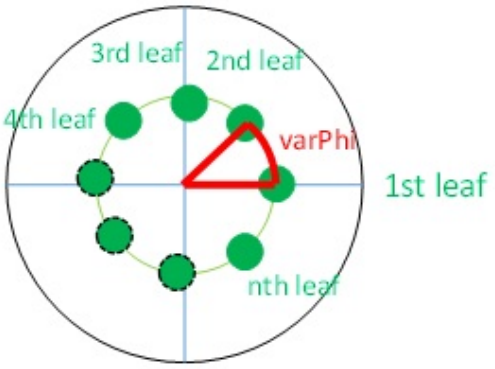

FIGURE 6.15: Schematic view of the assumed phyllotactic pattern of the rice plant

6.4 Optimization Model

The aim of the optimization model is to find the optimal value of the objective function which is the function calculating the value of light absorbed by the rice plant with combining fifteen selected parameters describing the plant structure. The constructed model of optimization itself consists of five classes that represent the methods of optimization (Figure 6.16). The classes are *FullFactorialDesign*, *SimpleRandomSampling*, *LatinHypercubeSampling*, *HillClimbing*, and *SimulatedAnnealing*; where they are part of the class *Optimization* with cardinality 1 - 1..0. Basically, all methods have several similar parameters: $parameterNumber$, $lowerBound$, $upperBound$, $incrementalValue$,

and *parameterType* which are identified as global parameters. Moreover, each class has local parameters used to run its functions, and the explanation of each class is delivered in further sub-sections.

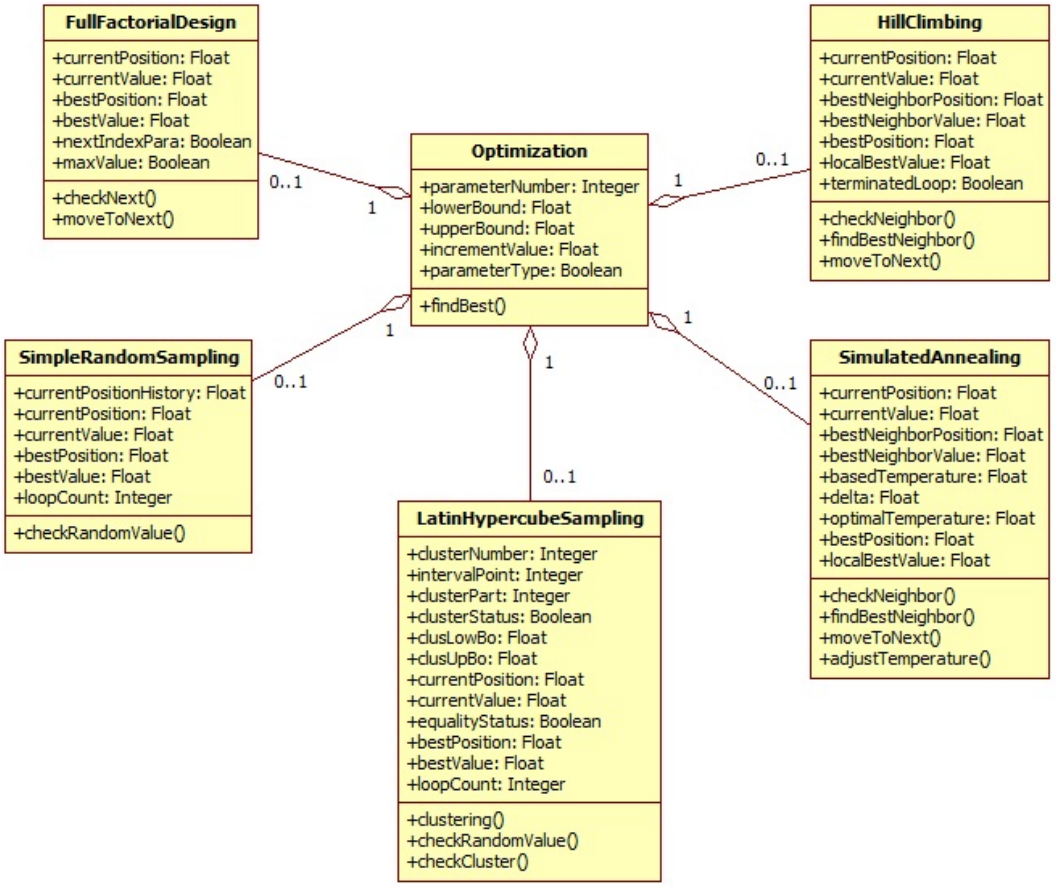

FIGURE 6.16: Class diagram of optimization model

The initiation process of global parameters is done in the procedure *Optimization*() (see Listing 6.10). Principally, five types of meta-parameters are initiated and defined here. The first is the parameter $paraNum$, the number of parameters of the optimization problem. It is defined here how many parameters will be used in the optimization process. For this research, as the purpose is to maximize the value of light intercepted by a plant through 15 selected structural parameters, the parameter $paraNum$ is defined as 15. The second one is the meta-parameter *paraType*. It is a binary indicator attached to each parameter; where 0 indicates that the parameter is a floating number, and 1 for a discrete one. This parameter is defined in array form, with parameter $paraNum$ (15) used as the number of array members. Furthermore, the next ones are the parameters $lowBo$, $upBow$, and $incVal$. Respectively, they are the lower bound, upper bound, and incremental value of the corresponding parameters. They are defined as array parameters with 15 members (or with $paraNum$ members).

Finally, the last part in the procedure *Optimization*() is a command for calling the procedure of optimization that is being used. All parameters that have been defined before are used to become the transferred parameters. For example, in Listing 6.10, the procedure *SimulatedAnnealing*() is called. Five (meta-)parameters that have been defined are transferred as input parameters for the further process of the procedure.

```
public void Optimization()
{
   int paraNum = 15;
   int[] paraType = {1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
      1};
   float[] lowBo = {1.0, 28.0, 0.0022, -0.3, 30.0, 15.0, 3.0,
      5.0, 1.0, 1.0, 0.68, 0.60, 0.022, -2.00, 90};
   float[] upBo =  {8.0, 92.0, 0.0045, 1.0, 45.0, 25.0, 12.0,
      23.0, 12.0, 10.0, 1.36, 0.75, 0.038, 2.00, 270};
   float[] incVal = {1.0, 1.0, 0.0001, 0.01, 1.0, 1.0, 1.0,
      1.0, 1.0, 1.0, 0.01, 0.01, 0.001, 0.01, 1};

   SimulatedAnnealing(paraNum, lowBo, upBo, incVal, paraType);
}
```

LISTING 6.10: XL code for procedure of optimization

The other command that is commonly used by all optimization methods except full-factorial design is a command part for randomizing the parameters value for the first time (Listing 6.11). This process is done in *paraNum* times. Two types of parameter distinguish the way to randomize. The command *irandom*() is used to randomly initiate the discrete value of parameters and *random*() is used to randomize the floating point value of parameters. The variable *cPos*[] indicates the "current position" in the parameter space defining the current values of the parameters that will be used in calculating the value of the objective function. The parameter *copyCPos*[] is used to duplicate *cPos*[] for other purposes, for checking the next neighbour (Listing 6.29) in one procedure used by the methods hill climbing and simulated annealing. Decimal formatting can be used then based on the specified purpose by using the command "$Float.valueOf(df.format(cPos[i]))$", where *df* is a defined decimal format. For example, the definition "$DecimalFormat\ df = new\ DecimalFormat$(".####");" is used to set four digits of value behind the decimal point.

```
for(int i = 0; i <= paraNum-1; i++)
{
   //for discrete value
   if(paraType[i] == 1)
   {
      cPos[i] = irandom((int)(lowBo[i]), (int)(upBo[i]));
      copyCPos[i] = cPos[i];
   }
   //for floating point value
   else
   {
      cPos[i] = random(lowBo[i], upBo[i]);
      //Decimal value formatting...
      cPos[i] = Float.valueOf(df.format(cPos[i]));
```

```
      copyCPos[i] = cPos[i];
   }
}
```

LISTING 6.11: XL code for randomizing parameters for the first time

The other XL code part used by all methods is the code for getting the value of the objective function. Conceptually, the model can accumulate the total light interception during modelled rice plant growth in the vegetative phase. The value of accumulated light interception is recorded in the variable $cVal$ by calling the function $plantGrow()$ (the objective function) and transferring parameters $paraNum$ and $cPos$ (Listing 6.12).

```
cVal = plantGrow(paraNum, cPos);
```

LISTING 6.12: XL code for calling the objective function plantGrow() and getting its value

In the following parts, the explanation of each sub-model is delivered. The procedure of each method in the form of complete pseudocode and specific parts in XL code are also given.

6.4.1 Full Factorial Design (FFD) Submodel

As mentioned before, there are five parameters used in this method. The first is $paraNum$. It is the number of parameters. How many parameters are used can be de defined here. For this research, the number of parameters is 15. The next ones are $lowBo[]$ and $upBow[]$. They are the lower bound and upper bound value of each parameter used. They are identified in array form. The last two are $incVal$ and $paraType$. They are respectively an incremental value and type (continuous or discrete) of parameters. They totally depend on the meaning of each parameter.

```
Procedure FullFactorialDesign(paraNum, lowBo[], upBo[],
   incVal[], paraType[])
Begin
   Variable definition...
   //initiating cPos
   For(h=0 upto paraNum-1, h++)
      cPos[h] <-- lowBo[h]
   End for

   //full factorial design looping
   While(maxVal=false)
      //getting result of objective function
      cVal <-- objFunction(paraNum, cPos)
      If(cVal > bestVal) //and comparing with the best
         bestVal <-- cVal
         For(i=0 upto paraNum-1, i++)
            bestPos[i] <-- cPos[i]
         End for
```

```
      End if

      incNextIndex <-- false
      For(j=0 upto paraNum-1, j++)
         //increasing cPos
         If(j=0)
            cPos[j] <-- cPos[j] + incVal[j]
         Else
            If(incNextIndex = true)
               cPos[j] <-- cPos[j] + incVal[j]
            Else
               break
            End if
         End if

         //resetting value at this index to low bound
         //if exceeded upper bound
         If(cPos[j]>upBo[j])
            cPos[j] <-- lowBo[j]
            incNextIndex <-- true
         Else
            incNextIndex <-- false
         End if
      End for

      //checking all parameters must less than upper bound
      //and setting true to maxVal then
      maxVal <-- maxValCheck(cPos, upBo, paraNum)
   End while

   //getting result of objective function
   cVal <-- objFunction(paraNum, cPos)

   If(cVal > bestVal) //and comparing with the best
      bestVal <-- cVal
      For(i=0 upto paraNum-1, i++)
         bestPos[i] <-- cPos[i]
      End for
   End if
End

Function maxValCheck(cPos[], upBo[], paraNum)
Begin
   eqVal <-- true
   For(i=0 upto paraNum-1, i++)
      If(cPos[i]!=upBo[i])
         eqVal <-- false
      End if
   End for
   return(eqVal)
End
```

LISTING 6.13: The pseudocode of the full factorial design (FFD) method with checking upper bound equality function

The Listing 6.13 is the detailed pseudocode of FFD. As we know FFD is a method to find the best value (of an objective function) by scanning all possibilities. By using the statement "$cPos[h] <-- lowBo[h]$", the lower bound values are defined as the fist parameter values that will be used, where $cPos$ indicates the current position in parameter space and h is the index of the parameter. To get the value of the objective function, the statement "$cVal <-- objFunction(paraNum, cPos)$" is used; where $cVal$ indicates the current value, and $objFunction()$ is the call of a model function to

produce the value of the objective function with transferring parameters $paraNum$ and $cPos$.

Furthermore, $cVal$ is compared with $bestVal$. The variable $bestVal$ is the currently best value, where for the first time it is initialized with zero. If $cVal$ is greater than $bestVal$, $cVal$ becomes $bestVal$, and all $bestPos$ will be replaced by $cPos$. For the next parameter values, the statement "$cPos[j] <-- cPos[j] + incVal[j]$" is executed, where $cPos$ is incremented by $incVal$ with index variable j. The process is continued then as long as $maxVal$ is false. The variable $maxVal$ is a variable used to record the boolean value true when the last parameter combination is found. The value of the variable $maxVal$ itself is checked by using the function $maxValCheck()$ in the statement "$maxVal <-- maxValCheck(cPos, upBow)$" by transferring parameters $cPos$, $upBo$, and $paraNum$.

The algorithm for moving to subsequent parameter combinations can be implemented in XL code (Listing 6.14). The step to move to the next parameter combination is conducted under the condition $while(maxVal == false)$ or until variable i has reached $paraNum - 1$. The variable $maxVal$ itself is coming from the function $maxValCheck()$ (Listing 6.15) that will return the boolean value true if all parameters reach their upper bound.

```
while(maxVal == false)
{
   boolean incrementNextIndex = false;

   for(int i=0; i<=paraNum-1; i++)
   {
      //increment current index i
      if(i==0) //at index 0, no dependency on flag
      {
         cPos[i] += incVal[i];
      }
      else //use flag to decide if increment is needed
      {
         if(incrementNextIndex)
         {
            cPos[i] += incVal[i];
         }
         else
         {
            break;
         }
      }

      //resetting value at this index to min if exceeded max
      if(cPos[i]>upBo[i])
      {
         cPos[i]=lowBo[i];
         incrementNextIndex=true;
      }
      else
      {
         incrementNextIndex=false;
      }
   }
   maxVal = maxValCheck(cPos, upBo, paraNum);
}
```

LISTING 6.14: XL code for moving to next parameter combination in full factorial design method

```
private boolean maxValCheck(float[] cPos, float[] upBo,
   int paraNum)
{
   boolean eqVal = true;
   for(int i=0; i<=paraNum-1; i++)
   {
      if(cPos[i]!=upBo[i])
      {
         eqVal = false;
      }
   }
   return (eqVal);
}
```

LISTING 6.15: XL code for checking the value of each parameter for having reached its upper bound or not

6.4.2 Simple Random Sampling (SRS) Submodel

Conceptually, SRS is different from FFD. It is used to find the optimal value of a lot of possibilities by randomizing. It chooses random parameter values used in calculating a value of the objective function. It can be executed many times based on how the variable *loopCount* is defined (see pseudocode in Listing 6.16). The parameters are here divided into two types (see parameter *paraType*). The type is used when defining the value of *cPos*. The statement "$cPos[i] <-- irandom(lowBo[i], upBow[i])$" is used to generate an integer random number, and the statement "$cPos[i] <-- random(lowBo[i], upBo[i])$" is used to generate a floating point random number. Both random values generated are confined to the interval between lower bound and upper bound value of each parameter.

All parameters that are randomized should not be equal to previous ones. For that reason, a 'parameter history table' is defined. All parameters that have been used are recorded in the history table by using the 2 dimensional array variable $cPosHis[][]$. It can record the index of parameter combination and value of all parameters. The process for checking equality can be done by checking each parameter of *cPos* (please see statement "$if\ CPosHis[j][k] = cPos[k]$" then "$eqCount++$"). Two parameter combinations are the same if the variable *eqCount* is equal to the number of parameters. It means, equality is found for each parameter.

```
Procedure SimpleRandomSampling(paraNum, lowBo[], upBo[],
incVal[], paraType[])
Begin
   Variable definition...
```

```
   While(loopCount)
      //initiating the first cPosHis with zero
      If(loopCount=0)
         For(h=0 upto h<=paraNum-1, h++)
            cPosHis[loopCount][h] <-- 0
         End for
      End if

      //randomizing new parameter and checking the equality
      randLoop <-- true
      While(randLoop=true)
      //randomizing new parameter
         For(i=0 upto i<=paraNum-1, i++)
            If(paraType[i]=1)//for integer number
               cPos[i] <-- irandom(lowBo[i], upBo[i])
            Else
               cPos[i] <-- random(lowBo[i], upBo[i])
            End if
         End for

         //checking the equality
         For(j=0 upto j<=loopCount, j++)
            eqCount <-- 0
            For(k=0 upto k<=paraNum-1, k++)
               If(cPosHis[j][k]=cPos[k])
                  eqCount++
               End if
            End for

            If(eqCount=paraNum)//finding equality
               randLoop <-- true
               j is terminated
            Else
               randLoop <-- false
            End if
         End for
      End while

      cVal <-- objFunction(paraNum, cPos)

      //checking the best value
      If(cVal > bestVal)
         bestVal <-- cVal
      End if

      //updating cPosHis
      loopCount++
      For(l=0 upto l<=paraNum-1, l++)
         cPosHis[loopCount][l] <-- cPos[l]
      End for
   End while
End
```

LISTING 6.16: Pseudocode of simple random sampling method

For checking the history table, the XL code in Listing 6.17 is used. It is used to make sure that the parameter combination obtained from the random choice was never used before. This algorithm is done as long as *loopCount* is less than how many times the choice is repeated (for example here, it is *loopNum* as 9 for ten times repetition). For the first time, the first member of the history table (the first *cPosHis*) is defined by zero and *randLoop* is defined by true, it is used to control it the random choice of parameters needs to be repeated again or not. In the second loop, the variable *eqCount* is needed to count how many parameters are equal with previous ones. If *eqCount* is equal to the number of parameters, the new parameters are all equal to the previous parameters, and

the random choice process has to be taken again. The history table is updated every time the procedure gets a new parameter combination.

```
loopNum = 9;

while(loopCount<=looNum)
{
   //initiating the first cPosHis with zero
   if(loopCount==0)
      for(int f=0; f<=paraNum-1; f++)
         cPosHis[loopCount][f] = 0;

   randLoop = true;
   while(randLoop == true)
   {
      //randomizing new parameters...
      ...

      //checking the parameters equality in history table
      for(int j=0; j<=loopCount; j++)
      {
         eqCount=0;
         for(int k=0; k<=paraNum-1; k++)
         {
            if(cPosHis[j][k] == cPos[k])
            {
               eqCount++;
            }
         }

         //quit checking and finding the equality
         if(eqCount==paraNum)
         {
            j=1000; //setting j more than loopCount
            randLoop = true;
         }
         else
         {
            randLoop = false;
         }
      }
   }

   //other commands...
   ...

   //updating cPosHis
   loopCount++;
   for(int l=0; l<=paraNum-1; l++)
   {
      cPosHis[loopCount][l] = cPos[l];
   }
}
```

Listing 6.17: XL code for checking history table

6.4.3 Latin Hypercube Sampling (LHS) Submodel

Firstly, the Figure 6.17 simply describes some concepts used in the method LHS. The simple example depicted in Figure 6.17 explains that the number of parameters ($paraNum$) is two, the number of intervals of parameter values ($clusterPart$) of the first and the second parameter are respectively 3 and 2, and the total number of clusters ($clusNum$) is 6.

The LHS is more complex than SRS. Principally, it is based on the SRS method, however several area cubes (clusters) of possibilities are made. The range of each parameter is divided into several parts, i.e., intervals ($clusPart[]$). The number of clusters is easily known now by multiplying all $clusPart[]$. In pseudocode (Listing 6.18), this can be seen in the statement "$clusNum <-- clusNum * clusPart[e]$", where e is an index of parameters. The parameter interval between lower and upper bound of each part can be calculated then by using the statement "$interVal[f] <-- (int)((upBo[f - lowBo[f])/clusPart[f])$" for integer parameter type, and "$interVal[f] <-- ((upBo[f - lowBo[f])/clusPart[f])$" for floating point parameter type. And the lower and upper bound of each parameter part are also determined by respectively using statements "$clusLowBo[g][h] <-- lowBo[g] + (h * interVal[g])$" and "$clusUpBo[f] <-- lowBo[j] + ((h + 1) * interVal[g])$".

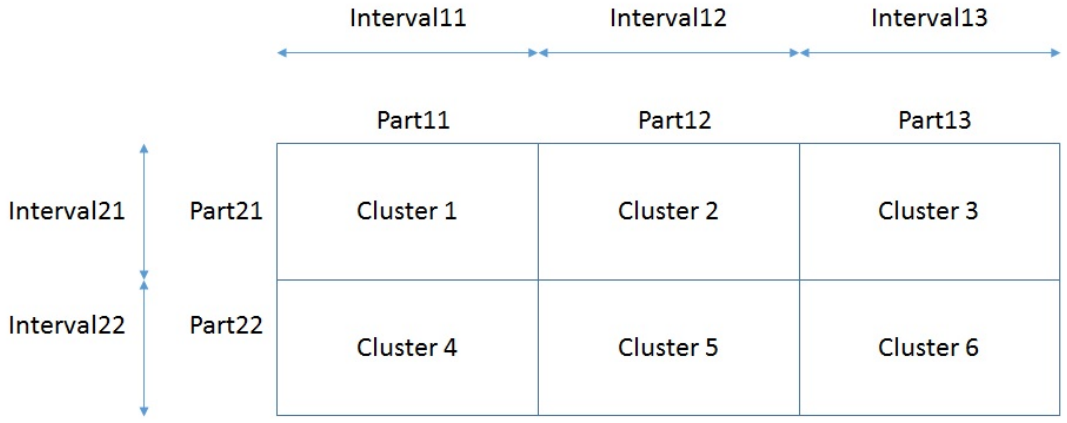

FIGURE 6.17: The cluster configuration in LHS of a simple example with two parameters and six clusters

The variable *eqStatus* is used to check equality status of clusters. The boolean value true indicates that the random choice of parameter values must be repeated. The variable *eqStatus* is determined from the function *checkCluster*(). In addition, for the wider looping, the variable *loopCount* is used. The variable type of *loopCount* is numeric (integer). It is used to control how many times the main repetition has to be conducted. For example, the total number of cluster is $clusNum$, and each cluster is desired to be visited at most 2 times (defined as $oneClusMaxNum$), the *loopCount* is defined as 2 $\times$ $clusNum$.

```
Procedure LatinHypercubeSampling(paraNum, lowBo[], upBo[],
incVal[], clusPart[], paraType[])
Begin
   Variables definition...
```

```
   //calculating number of cluster
   clusNum <-- 1
   For(e=0 upto paraNum-1, e++)
      clusNum <-- clusNum * clusPart[e]
   End for

   //calculating interval
   For(f=0 upto paraNum-1, f++)
      If(paraType[f]=1)//for integer number
         interVal[f] <-- (int)((upBo[f]-lowBo[f])/clusPart[f])
      Else
         interVal[f] <-- (upBO[f]-lowBo[f])/clusPart[f]
      End if
   End for

   //setting lowerbound and upperbound in every part
   For(g=0 upto paraNum-1, g++)
      For(h=0 upto clusPart[g]-1, h++)
         clusLowBo[g][h] <-- lowBo[g] + (h*interVal[g])
         clusUpBo[g][h] <-- lowBo[j] + ((h+1)*interVal[g])
      End for
   End for
   While(loopCount)
      //randomizing new parameter and checking the equality
      eqStatus <-- true
      While(eqStatus=true)
         //randomizing new parameter
         For(i=0 upto i<=paraNum-1, i++)
            If(paraType[h]=1)//for integer number
               cPos[i] <-- irandom(lowBo[i], upBow[i])
            Else
               cPos[i] <-- random(lowBo[i], upBow[i])
            End if
         End for

         //checking cluster equality status
         eqStatus <-- checkCluster(paraNum, clusPart, cPos,
                loopCount, clusLowBo, clusUpBo, clusNum)
      End while

      cVal <-- objFunction(paraNum, cPos)
      //checking the best value
      If(cVal > bestVal)
         bestVal <-- cVal
      End if

      loopCount++
   End while
End
```

LISTING 6.18: Pseudocode of latin hypercube sampling method

The XL code for calculating the interval is shown in Listing 6.19. In addition, setting the limits (lower and upper bound) of each part is represented in Listing 6.20. They are $clusLowBo[][]$ and $clusUpBo[][]$, and defined as two dimensional arrays; where the first index represents the parameter (see Figure 6.18 for an example). The number of cluster ($clusNum$) can be calculated by using equation (6.33), where k is indexing the parameters and $paraPart$ is the number of parts of each parameter.

$$clusNum = \Pi_{k=1}^{n} paraPart_k \tag{6.33}$$

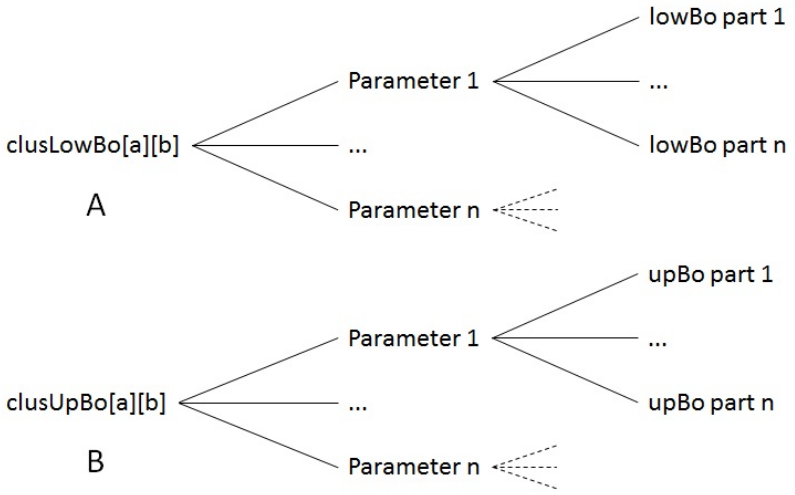

FIGURE 6.18: Structure preview of cluster lower bound (A) and cluster upper bound (B)

For checking the status of a cluster, the function *checkCluster*() is called (see Listing 6.21), and technically it returns the boolean value of equality status (*eqStatus*). Seven parameters are used here, they are: number of parameter, cluster part, current position, loop count number, lower bound of cluster, upper bound of cluster, and number of cluster.

```
for(int i=0; i<=paraNum-1; i++)
{
   if(paraType[i]==1)
   {
      interVal[i] = (int)((upBo[i]-lowBo[i])/clusterPart[i]);
   }
   else
   {
      interVal[i] = (upBo[i]-lowBo[i])/clusterPart[i];
   }
   print(interVal[i]+" ");
}
```

LISTING 6.19: XL code for setting the interval for each parameter

```
for(int j=0; j<=paraNum-1; j++)
{
   for(int k=0; k<=clusterPart[j]-1; k++)
   {
      clusLowBo[j][k] = lowBo[j] + (k*interVal[j]);
      if(k>0)
      {
         if(paraType[j]==1)
         {
            clusLowBo[j][k] = clusLowBo[j][k]+1;
         }
         else
         {
            clusLowBo[j][k] = clusLowBo[j][k]+0.1;
         }
      }
```

```
        if(paraType[j]==1)
        {
            clusUpBo[j][k] = clusLowBo[j][k] + interVal[j];
        }
        else
        {
            clusUpBo[j][k] = lowBo[j] + ((k+1)*interVal[j]);
        }
    }
}
```

LISTING 6.20: XL code for setting the lower bound and upper bound for each cluster

```
eqStatus=checkCluster(paraNum, clusPart, cPos, loopCount,
    clusLowBo, clusUpBo, clusNum);
```

LISTING 6.21: XL code for calling the function CheckCluster()

Another important variable is the cluster status (*clusStatus*[]). It is defined in array format with a number of array members which is equal to the number of clusters, and used to define the cluster status. In the function *checkCluster*(), for the first time, the cluster status is defined as zero (see Listing 6.22). If the value of cluster status of one cluster is equal to or more than the maximal number in one cluster (*oneClusMaxNum*), the function *checkCluster*() will return the boolean *equality* value true, and the cluster status will be decreased by one (Listing 6.23), because the process of random choice of parameter values must be repeated to find another parameter combination in another cluster. The parameter *oneClusMaxNum* is defined based on how many times one cluster is allowed to be visited by the parameter combination.

Another variable is *statusCounter*, it is used to count the status by incrementing *statusCounter* when one chosen random parameter value is in "interval of parameter" (please see again Figure 6.17). One parameter combination will be in one cluster (by adding *clusStatus*[]), if it has a *statusCounter* that is equal to *paraNum* (see Listing 6.24); where *count* is a counter of the clusters, and *statusCounter* will be started from 0 again.

```
if(loopCount == 0)
{
    for(int c = 0; c<=clusNum-1; c++)
    {
        clusStatus[c] = 0;
    }
}
```

LISTING 6.22: XL code for initiating the status of clusters

```
for(int r=0; r<=clusNum-1; r++)
{
    if(clusStatus[r]>=oneClusMaxNum)
    {
        equality = true;
        clusStatus[r]--;
```

```
    }
}
```

LISTING 6.23: XL code for checking and comparing the cluster status with a maximal number in one cluster and the function CheckCluster() will return the value true

```
if(statusCounter==paraNum)
{
   clusStatus[count]++;
   statusCounter=0;
}
else
{
   statusCounter=0;
}
```

LISTING 6.24: XL code for checking statusCounter and incrementing clusStatus

For the first time, it must be checked for the first part of each parameter (cluster 1, see Figure 6.17). If the condition is fulfilled, the *statusCounter* of the first cluster is incremented by one. If the *statusCounter* is equal to the number of parameters (*paraNum*), it will increment *clusStatus*[] that is arranged in new indexing; it means the location of all random combinations now are in cluster 1, the status of the first cluster is one now, and *statusCounter* is 0 again (see Listing 6.25).

```
for(int d=0; d<=paraNum-1; d++)
{
   if((cPos[d]>=clusLowBo[d][0])&&(cPos[d]<=clusUpBo[d][0]))
   {
      statusCounter++;
   }
}

if(statusCounter==paraNum)
{
   clusStatus[count]++;
   statusCounter=0;
}
else
{
   statusCounter=0;
}
```

LISTING 6.25: XL code for incrementing the status counter if the chosen random value is in the first interval of the parameter

For the next step, part by part of each parameter has to be checked, which part contains the value and which cluster visited can be identified then (by incrementing the cluster status). For checking, the algorithm in Listing 6.26 is used. Here, the pointer of interval (*intervalPoint*) is needed to point the index of the part when checking the lower and upper bound of each part for each parameter. To get the boolean value of *partLimit*, the function *valuesEqual*() is used by transferring the number of parameters, interval point, and cluster part (Listing 6.27).

```
while(partLimit == false)
{
   count++;
   boolean incrementNextIndex = false;
   for(int i=0; i<=paraNum-1; i++)
   {
      //increment current index i
      if(i==0) //at index paraNum, no dependency on flag
      {
         intervalPoint[i]++;
      }
      else //use flag to decide if increment is needed
      {
         if(incrementNextIndex)
         {
            intervalPoint[i]++;
         }
         else
         {
            break;
         }
      }

      //reset value at this index to min if exceeded max
      if(intervalPoint[i]>clusPart[i]-1)
      {
         intervalPoint[i] = 0;
         incrementNextIndex=true;
      }
      else
      {
         incrementNextIndex=false;
      }
   }

   for(int j=0; j<=paraNum-1; j++)
   {
      if((cPos[j]>=clusLowBo[j][intervalPoint[j]])&&
      (cPos[j]<=clusUpBo[j][intervalPoint[j]]))
      {
         statusCounter++;
      }
   }

   if(statusCounter==paraNum)
   {
      clusStatus[count]++;
      statusCounter=0;
   }
   else
   {
      statusCounter=0;
   }
   partLimit = valuesEqual(paraNum, intervalPoint, clusPart);
}
```

Listing 6.26: XL code for checking part by part by using intervalPoint

```
private boolean valuesEqual(int paraNum, int[] intervalPoint,
int[] clusPart)
{
   boolean partLimit = true;

   for(int i=0; i<=paraNum-1; i++)
   {
      if(intervalPoint[i]!=clusPart[i]-1)
         partLimit = false;
   }
   return (partLimit);
}
```

Listing 6.27: XL code for checking the limit of an interval

6.4.4 Hill Climbing (HC) Submodel

In the method HC, that the optimal value seeking will be stopped if the method finds one value where there is no neighbour which has a better value (see pseudocode in Listing 6.28). To define the value *cPos* firstly, a random choice of the parameter value is done. The value of parameters will be in between their lower bound and upper bound values (same as the algorithm done in SRS and LHS methods).

```
Procedure HillClimbing(paraNum, lowBo[], upBo[], incVal[],
paraType[])
Begin
   Variable definitions...

   //randomizing new parameter
   For(i=0 upto i<=paraNum-1, i++)
      If(paraType=1) //for integer number
         cPos[i] <-- irandom(lowBo[i], upBow[i])
      else
         cPos[i] <-- random(lowBo[i], upBow[i])
      End if

      copyCPos[i] <-- cPos[i]
   End for

   cVal <-- objFunction(paraNum, cPos)

   //looping until local optimum is found
   While(finish is not terminated)
      For(j=-1 upto j<=1, j=j+2) //initiating previous and next
         For(k=0 upto k<=paraNum-1, k++)
            cPos <-- copyCPos
            cPos[k] <-- cPos[k] + (j*incVal[k])
            //checking parameter range
            If((cPos[k]>=loBo[k])&&(cPos[k]<= upBo[k]))
               nVal <-- objFunction(paraNum, cPos)
            End if

            //finding the best neighbor
            If(nVal>=bestN)
               bestN <-- nVal
               bestNPos <-- cCPos
            End if
         End for
      End for

      //finding local optimum
      If(cVal>=bestN)
         Local optimum is found
         Finish is terminated
      Else //move to next position
         cVal <-- bestN
         cPos <-- bestNPos
         copyCPos <-- cPos
      End if
   End while
End
```

LISTING 6.28: Pseudocode of hill-climbing method

To initiate the index and value of neighbours (previous and next neighbours), a *nested* for loop is used. The first for is used to determine the index of a neighbour (see statement "$For(j = -1\ upto\ k <= 1,\ j = j + 2)$"), and the second for is used to

identify the value of the neighbour (see statement "$For(k = 0\ upto\ k <= paraNum - 1, k++)$"). The variable $nVal$ means a neighbour value, it is obtained from the objective function. And $nVal$ is compared with $bestN$, where $bestN$ is the currently best value of a neighbour. For the first time, $bestN$ could be a very small value (for example -999999). The optimal value seeking is continued by setting $cVal$ as $bestN$ ("$cVal <-- bestN$"), if $bestN$ is greater than $cVal$. And the process is terminated when $cVal$ is greater than $bestN$, it means the local optimum is found.

Another important part used by the method hill climbing (also for the next method simulated annealing) is the procedure to check the neighbours' parameters and a value of the objective function. The implementation of this part is shown in Listing 6.29. The previous and next indexes of the neighbour (for each parameter) can be traced by using the statement "$for(int\ j = -1;\ j <= 1;\ j = j + 2)$"; where both indexes are defined by the statement "$cPos[k] = cPos[k] + (j * incVal[k])$;". If the index is in the range between lower bound and upper bound, the parameter $nVal$ (value of neighbour) will store the value of the objective function. If it is out of the range, the variable $nVal$ will be 0.

The variable $nVal$ will be compared with $bestN$ to find the best value of a neighbour. The variable $bestN$ itself could be defined as a very small value for the first time. If $nVal > bestN$, $bestN = nVal$ then; and all neighbour's parameters are replaced by $cPos$ ($bestNPos[] = cPos[]$).

```
for(int j=-1; j<=1; j=j+2)
{
   for(int k = 0; k<= paraNum-1; k++)
   {
      for(int l=0; l<=paraNum-1; l++)
      {
         cPos[l] = copyCPos[l];
      }

      cPos[k] = cPos[k] + (j*incVal[k]);

      if((cPos[k] >= lowBo[k])&&(cPos[k]<=upBo[k]))
      {
         nVal = objFunction(paraNum, cPos);
      }
      else
      {
         nVal = 0;
      }

      if(nVal > bestN)
      {
         bestN = nVal;
         for(int i=0; i<=paraNum-1; i++)
         {
            bestNPos[i] = cPos[i];
         }
      }
   }
}
```

Listing 6.29: XL code for checking next neighbours' parameters and value

After all neighbours are checked, and the best value of a neighbour has been obtained as well; it will be compared with the current value ($cVal$). If $cVal$ is better than $bestN$, it means that the local optimum has been found, the program loop process is terminated (with giving the boolean value true to *terminatedLoop*). However, if $bestN$ is better than $cVal$, the loop process is continued to the next group of parameters (the next parameter combination) by setting $cVal$ as $bestN$ and replacing all current parameters with parameters of the neighbour (Listing 6.30).

```
while (terminatedLoop == false)
{
   Other command lines....
   if(cVal >= bestN) //getting the local optimum
   {
      terminatedLoop = true;
   }
   else //move to next parameter (best neighbour) as current value
   {
      cVal = bestN;
      bestN = 0;
      for(int i=0; i<= paraNum-1; i++)
      {
         cPos[i] = bestNPos[i];
         copyCPos[i] = cPos[i];
      }
   }
}
```

LISTING 6.30: XL code for getting the local optimum in the hill climbing method

6.4.5 Simulated Annealing (SA) Submodel

Simulated annealing (SA) is similar to hill climbing. The local optimum is found if the search process has reached the peak of the hill, and the process of optimum seeking is stopped. However, in SA, the process has a probability to be continued. Here, the variable *delta* is used ("$delta <-- bestN$ - $cVal$"); and it is used to get the probability from the result of a comparison. The probability is coded by using the if condition ("$If(exp(-delta/baseTemp) > random(0..1))$"). The process will be continued when the condition value is true (Listing 6.31).

The variable $baseTemp$ is a control parameter called "adjusted temperature" used to be an indicator to practically control the simulated annealing process. The process will be terminated if the value of the variable $baseTemp$ reaches the value of the variable $optTemp$, where $optTemp$ is an optimal temperature that has been defined before.

```
Procedure SimulatedAnnealing(paraNum, lowBo[], upBo[], incVal[],
   paraType[])
Begin
   Variable definitions...
   baseTemp <-- 100
```

```
   optTemp <-- 40

   //randomizing new parameter
   For(i=0 upto i<=paraNum-1, i++)
      If(paraType[i]=1)
         cPos[i] <-- irandom(lowBo[i], upBow[i])
      Else
         cPos[i] <-- random(lowBo[i], upBow[i])
      End if
      copyCPos[i] <-- cPos[i]
   End for

   cVal <-- objFunction(paraNum, cPos)

   //looping until local optimum is found
   While(baseTemp>=optTemp)
      For(j=-1 upto j<=1, j=j+2) //initiating previous and next
         For(k=0 upto k<=paraNum-1, k++)
            cPos <-- copyCPos
            cPos[k] <-- cPos[k] + (j*incVal[k]);
            //checking parameter range
            If((cPos[k]>=loBo[k])&&(cPos[k]<= upBo[k]))
               nVal <-- objFunction(paraNum, cPos)
            End if

            //finding the best neighbor
            If(nVal>=bestN)
               bestN <-- nVal
               bestNPos <-- cPos
            End if
         End for
      End for

      //finding local optimum
      If(bestN>=cVal>)
         cVal <-- bestN
         cPos <-- bestNPos
         copyCPos <-- cPos
         bestN <-- 0
      Else
         delta <-- bestN - cVal
         //Probability
         If(exp(-delta/baseTemp) > random(0..1))
            cVal <-- bestN
            cPos <-- bestNPos
            copyCPos <-- cPos
            bestN <-- 0
         End if
      End if

      //adjuting baseTemp
      adjust(baseTemp)
   End while
End
```

Listing 6.31: Pseudocode of simulated annealing method

The XL code implementation in Listing 6.29 can be used for checking the next neighbours' parameters. After all neighbours are checked, and the best value of a neighbour ($bestN$) has been obtained as well; it will be compared with the current value ($cVal$). If $bestN$ is better than $cVal$, it means the process has to be continued to the next step, with replacing the current value and parameters with the neighbour's value and position. However, if the other condition occurs ($cVal$ is better than $bestN$), actually the local optimum value is found, and the process terminates.

However, in the method simulated annealing, there is still the possibility to continue, if the adjusted temperature (*baseTemp*) has not reached the optimum temperature (*optTemp*) and the probability condition is fulfilled. The probability can be taken into account by calculating the variable *delta* ("$delta = bestN - cVal$") and comparing "$Math.exp(-delta/baseTemp)$" with a random value ($randVal$) (Listing 6.32). The process will really terminate if *baseTemp* is less than *optTemp*.

```
while (baseTemp >= optTemp)
{
   if(bestN >= cVal) //move to the next parameters
   {
      cVal = bestN;
      for(int i=0; i<= paraNum-1; i++)
      {
         cPos[i] = bestNPos[i];
         copyCPos[i] = cPos[i];
      }
   }
   else //checking possibility to move to the next parameters
   {
      randVal = random();
      delta = bestN - cVal;
      if(Math.exp(-delta/baseTemp)>randVal)
      {
         cVal = bestN;
         for(int i=0; i<= paraNum-1; i++)
         {
            cPos[i] = bestNPos[i];
            copyCPos[i] = cPos[i];
         }
      }
   }

   baseTemp = baseTemp * 0.95;
}
```

LISTING 6.32: XL code for getting the local optimum and calculating delta for checking probability condition

Chapter 7

The Result of Model and Simulation

7.1 Rice Plant Morphology Comparisons

The constructed rice plant model produces several important data and information. Two graphical informations about leaf length and width (in full growth) of four Indonesian rice varieties that have been successfully produced by the constructed model are delivered by Figures 7.1 and 7.2. The graphs in Figure 7.1 say that the variety Fatmawati has the longest maximum length of leaf, where the leaf rank number four from the top is the longest leaf [104] [167]. Figure 7.2 shows that the variety Fatmawati also has the widest leaves in every leaf rank. Indeed, these graphs were derived from the main empirical data [146], especially taking into account the maximum leaf length and width, by using [189] model (especially equations 6.1 upto 6.8, please see the subsection 6.1.1.1) and fitting several parameters, such as daily temperature, coefficients a_1 and a_2, the coefficient of final leaf length, etc. The fitting process was done to produce the model of "normal shape" of four Indonesian rice plants based on the empirical data that we have.

Furthermore, the variety Fatmawati has the longest internode at every internode rank as well. Figure 7.3 shows the internode length for each rank at the main stem of four Indonesian rice varieties. This information is produced by running the model in full growth simulation. The sum of all internode lengths is equal to the maximum length of the stem. For the length of the stem, the variety Fatmawati also has the longest stem both at the day of the end of the vegetative phase and after finishing growth (see Figure 7.4). Those curves were obtained from the empirical data [146] by using the model in equation (6.28) and fitting the parameter "internode coefficient" (see subsection 6.1.2.1).

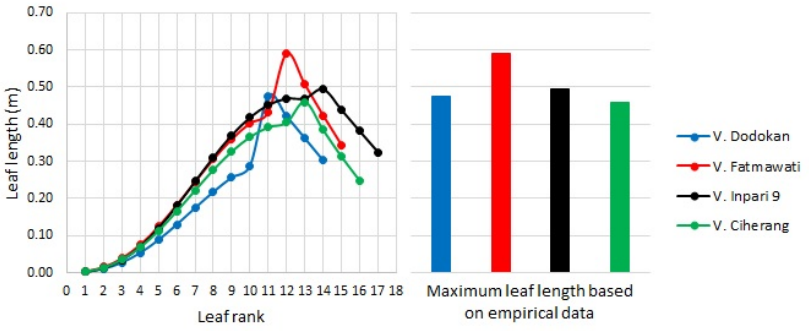

FIGURE 7.1: Graph of the leaf length at the main stem of four Indonesian rice varieties

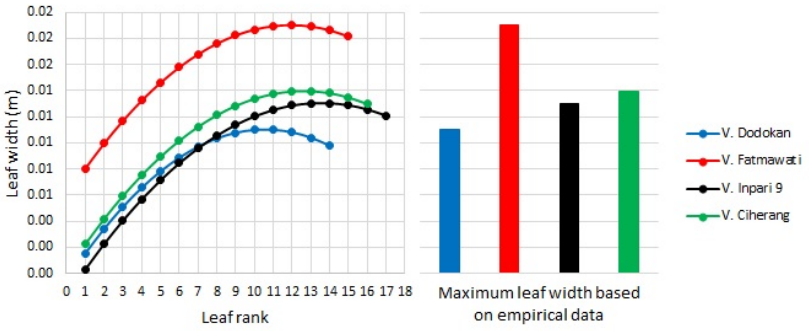

FIGURE 7.2: Graph of the leaf width at the main stem of four Indonesian rice varieties

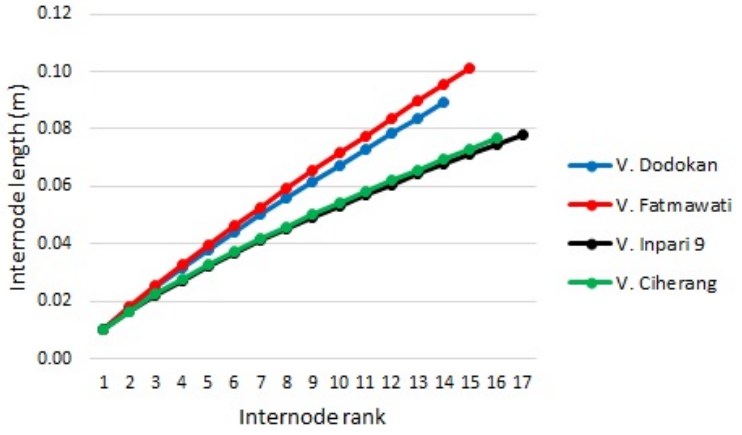

FIGURE 7.3: Graph of the internode length at the main stem of four Indonesian rice varieties

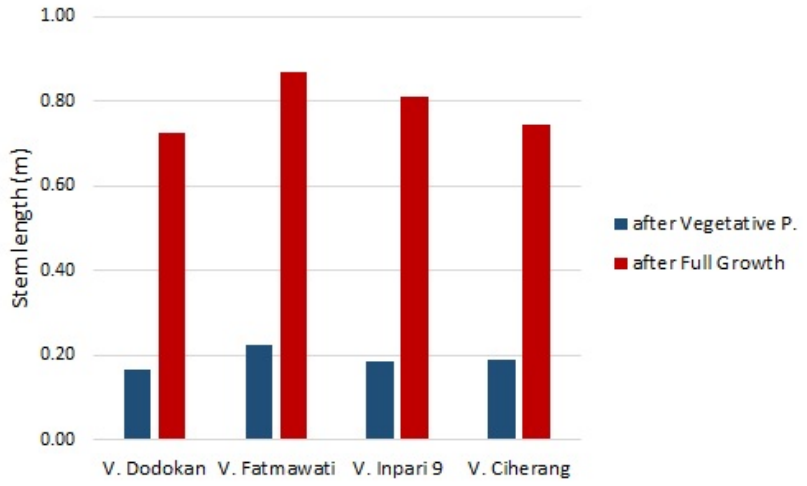

FIGURE 7.4: Graph of the length of the main stem of four Indonesian rice varieties (after vegetative phase and after finishing growth)

Furthermore, the model also successfully produces the total leaf area at the main stem of four Indonesian rice varieties both after the vegetative phase and after finishing growth (Figure 7.5). Here, the graphs also shows that the variety Fatmawati has the widest leaf area.

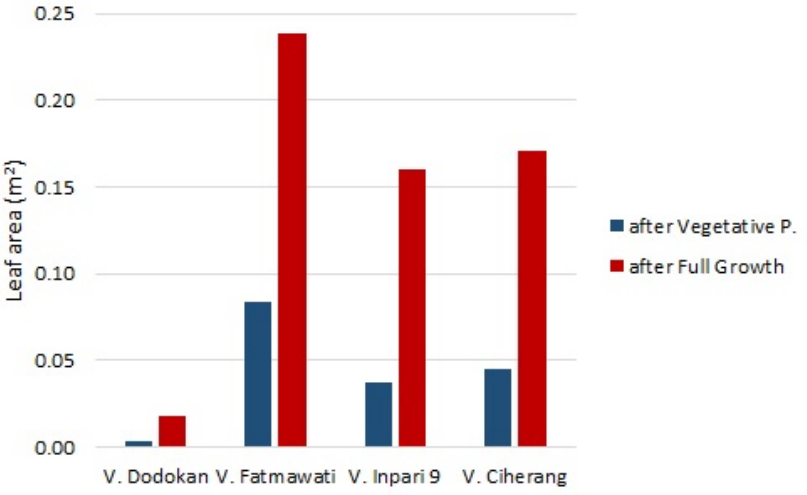

FIGURE 7.5: Graph of the leaf area at the main stem of four Indonesian rice varieties (after vegetative phase and after finishing growth)

Moreover, based on the empirical data (Table 3.1), the projected view of the simulated 3d-structure of four Indonesian rice plant varieties is shown in Figure 7.6. It shows the vegetative green organ only (stems and leaves), but in after finishing growth. It can be seen here that the variety Fatmawati is the highest one, it has the longest stem actually,

although it does not have the longest growth duration. The Figure also displays that the variety Inpari 9 is the lushest one, because it has the most productive tillers and leaf numbers. In addition, Figure 7.7 presents the histogram of relative accumulated light interception (RALI) of those four Indonesian rice varieties. Here, the Figure 7.7 shows that the variety Inpari 9 has the highest light interception (at the end of the vegetative phase).

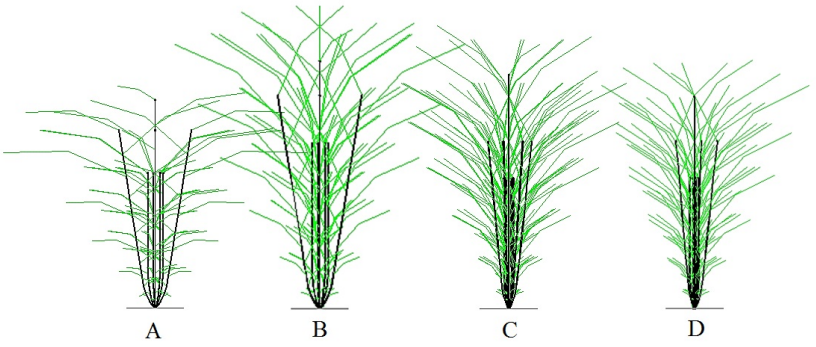

FIGURE 7.6: Schematic view of the model for four Indonesian rice pant varieties (after finishing growth). (A) Variety Dodokan (B) Variety Fatmawati (C) Variety Inpari 9 (D) Variety Ciherang

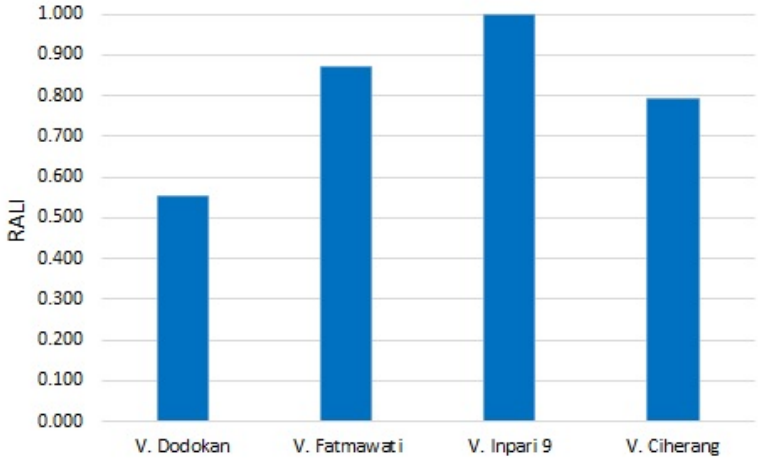

FIGURE 7.7: Histogram of Relative accumulated light interception (RALI) of four Indonesia rice varieties (at the end of the vegetative phase)

7.2 Sensitivity Analysis

Relative accumulated light interception (RALI) is an indicator to see the result of the sensitivity analysis for the 15 parameters which we selected at first for optimization and 6 additional parameters of plant structure before they are used in the optimizing process.

In this analysis, the variety Dodokan is exemplarily used as an object, and the assumed value of each variety Dodokan's parameter is an assumed value used in the simulation to model the "normal shape Dodokan" rice plant.

For the first parameter "starting day to bend" (Figure 7.8), the result indicates that if the leaves start to bend earlier, the plant will absorb more light; even though its pattern occurs in a narrow range of values (see Figure 7.8A). The graph is not purely linearly decreasing (see the detailed plot with rescaled y-axis in Figure 7.8B), it is probably affected by self-shadowing; as the light is not only coming from one point.

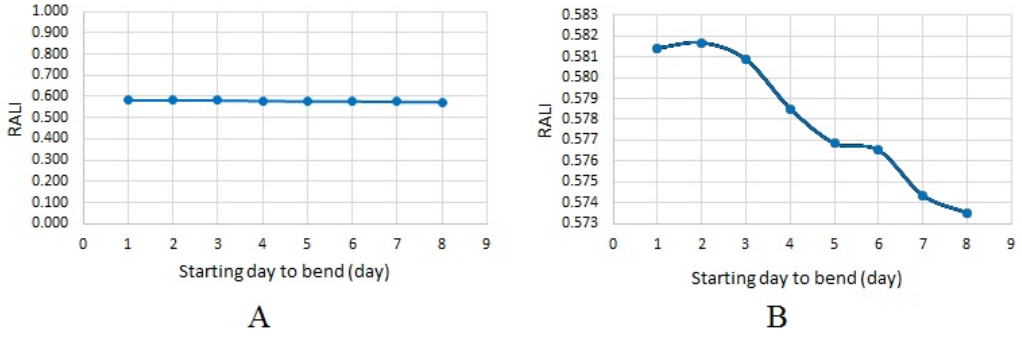

FIGURE 7.8: Graph of the sensitivity analysis result for parameter "starting day to bend". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 1.

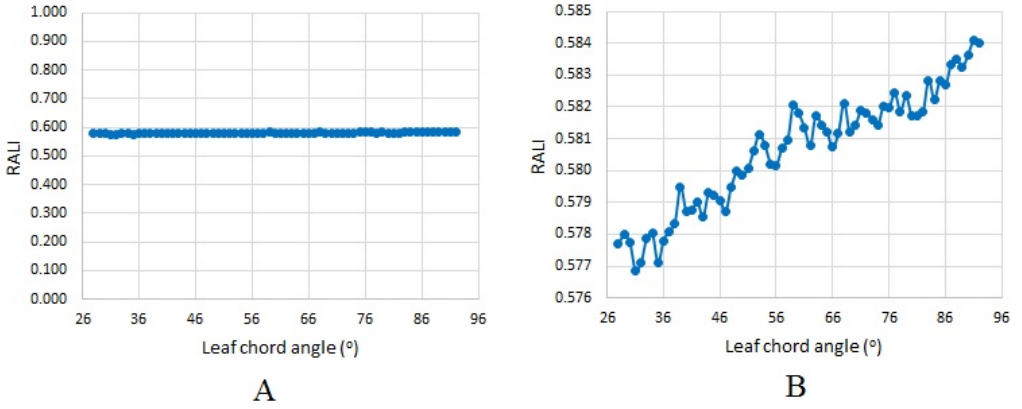

FIGURE 7.9: Graph of the sensitivity analysis result for parameter "leaf chord angle". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 70^o.

The result of sensitivity analysis for the second parameter "leaf chord angle" (Figure 7.9) produces an increasing rugged graph pattern (see Figure 7.9B) but it happens in a small range of RALI values (Figure 7.9A). A different pattern is exhibited by the third parameter "stem diameter", it strongly affects the RALI linearly (see Figure 7.10);

where the plant that has the larger diameter of stem will absorb much more light. This is reasonable, since the green organ stem with a larger diameter has a wider area. In addition, a fluctuating and declining trend in a narrow range occurs in the result of the sensitivity analysis for the fourth parameter "leaf bending coefficient" (Figure 7.11).

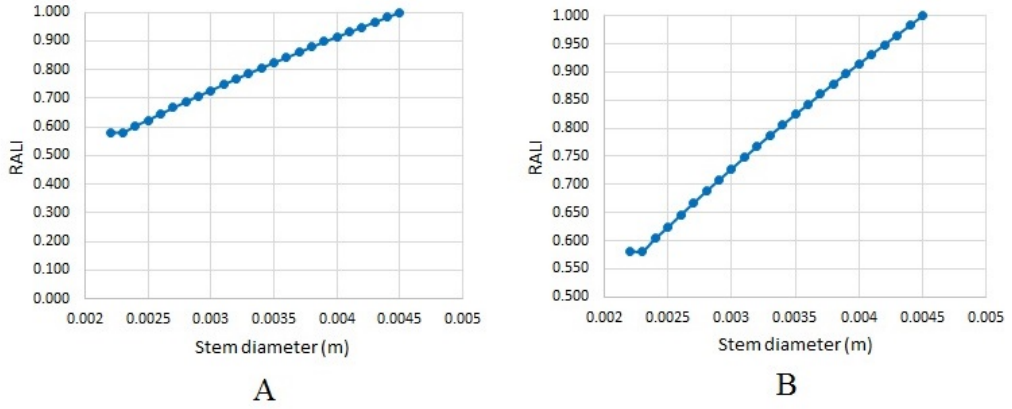

FIGURE 7.10: Graph of the sensitivity analysis result for parameter "stem diameter". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 0.0022m.

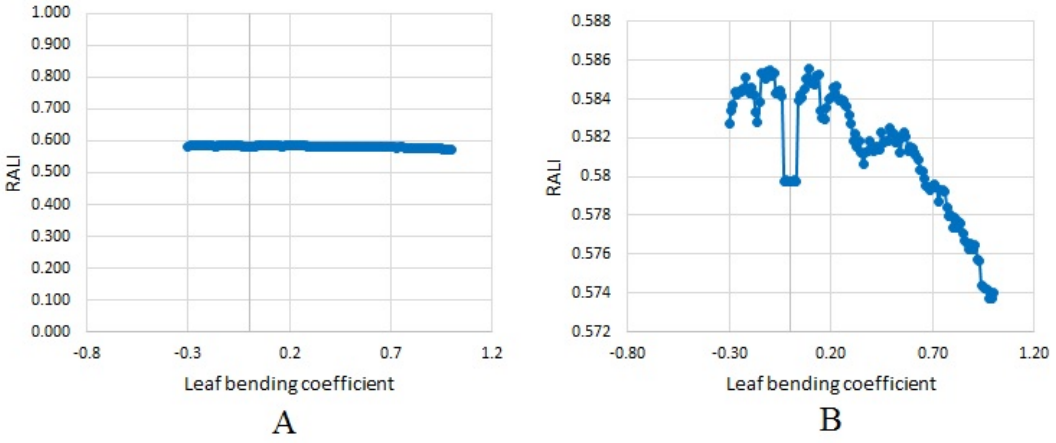

FIGURE 7.11: Graph of the sensitivity analysis result for parameter "bending coefficient". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 0.40.

The intercepted light value (RALI) grows nearly linearly in line with the increase of the "primary tiller angle" (see Figure 7.12). This is the result of the fifth parameter sensitivity analysis, although the increase takes place in a tight range. A similar graph shape also appears in the result of the sensitivity analysis for the next parameter "secondary tiller angle" (see Figure 7.13). A similar chart form but in a much wider range appears as well in the sensitivity analysis result of both parameters "primary tiller number" and "secondary tiller number" (Figure 7.14 and 7.15). This indicates that a rice plant with

more tillers, that consequently has more green organs, will be able to intercept much more light.

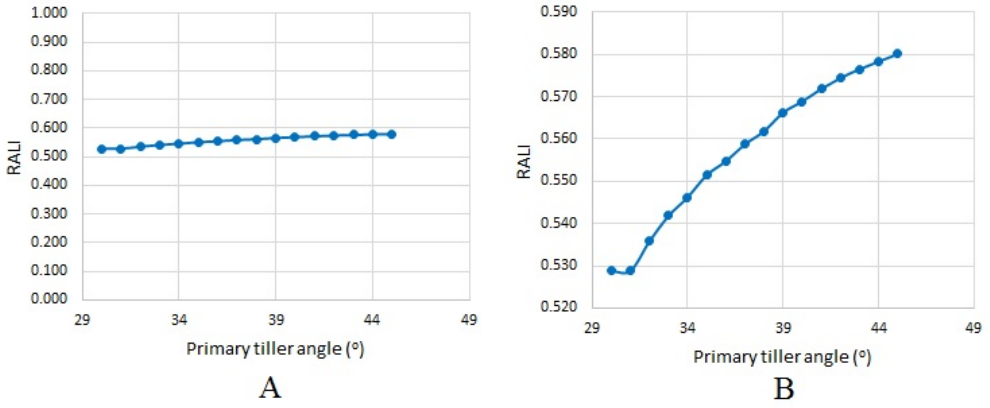

FIGURE 7.12: Graph of the sensitivity analysis result for parameter "primary tiller angle". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 45^o.

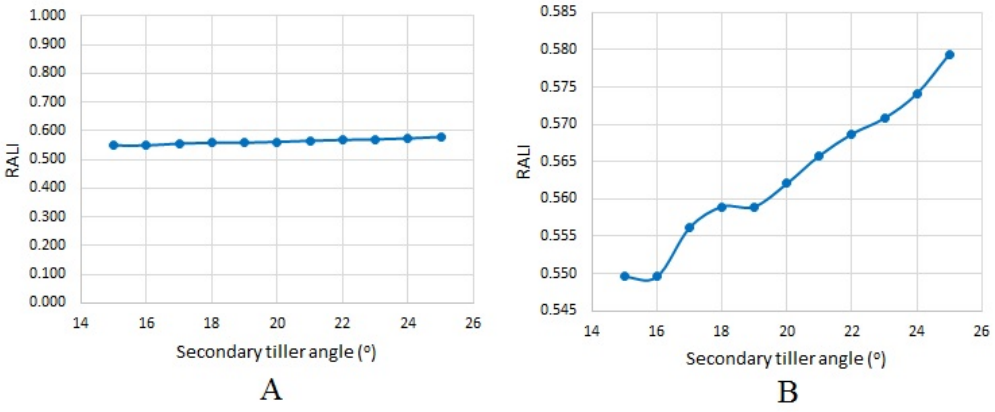

FIGURE 7.13: Graph of the sensitivity analysis result for parameter "secondary tiller angle". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 25^o.

The results of the sensitivity analysis for the parameters "internode number of primary tillers" (Figure 7.16) and "internode number of secondary tillers" are similar to each other (Figure 7.17). They increase linearly in the first few values and remain constant then. However, a little wider range appears in the result of the analysis for the parameter "internode number of secondary tillers". On the other hand, although a much wider range occurs in the result of the sensitivity analysis for the parameter "stem length coefficient", an increasing pattern also is the result of the analysis for both parameters "stem length coefficient" and "length proportion leaf/stem". However, the light interception values increase linearly and smoothly (see Figure 7.18) and with fluctuations (see Figure 7.19) respectively.

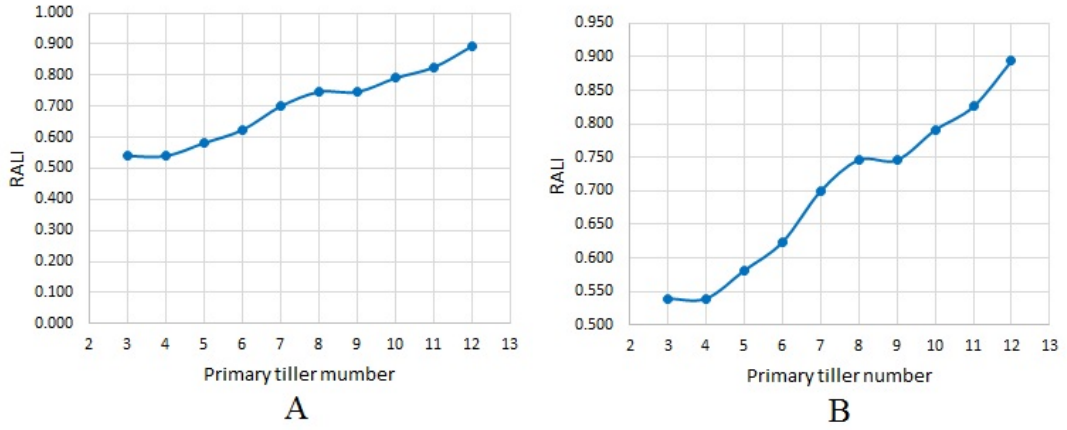

FIGURE 7.14: Graph of the sensitivity analysis result for parameter "primary tiller number". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 4.

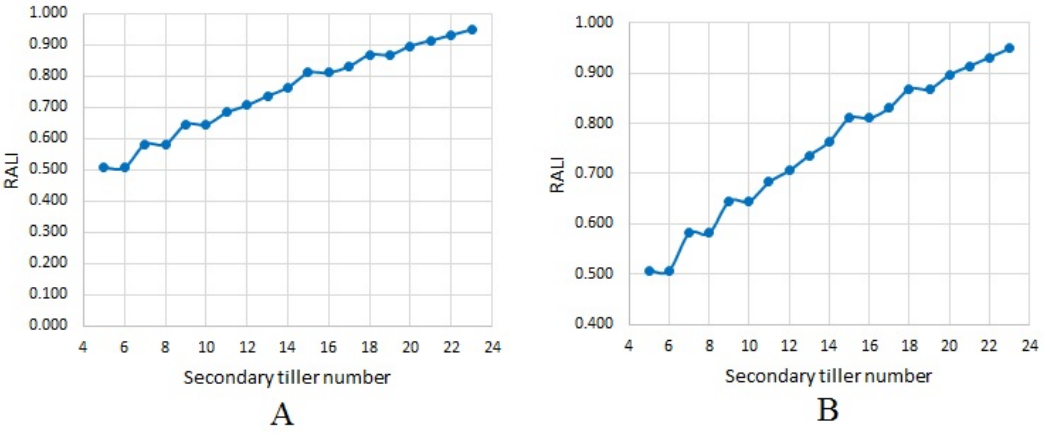

FIGURE 7.15: Graph of the sensitivity analysis result for parameter "secondary tiller number". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 7.

An increasing graph pattern occurs as well in the sensitivity analysis result for the parameter "width/length proportion of leaf" (Figure 7.20). A higher proportion here means that the plant has a wider leaf. It is realistic that a plant with wider leaves will be able to intercept more light. Indeed, the graph pattern of the sensitivity analysis result of the parameter "width/length proportion of leaf" seems ready constant (Figure 7.20A), however the increasing pattern can be seen clearly through the narrower range view in Figure 7.20B. Finally, a fluctuating graph pattern, within a very restricted range of values, occurs in the sensitivity analysis result for both parameters "delta chord angle" (Figure 7.21) and "phyllotaxy angle" (Figure 7.22). The narrower range views in Figure 7.21B and 7.22B can be used to get a clearer view of the fluctuating graph pattern of the analysis result for both parameters "delta chord angle" and "phyllotaxy angle".

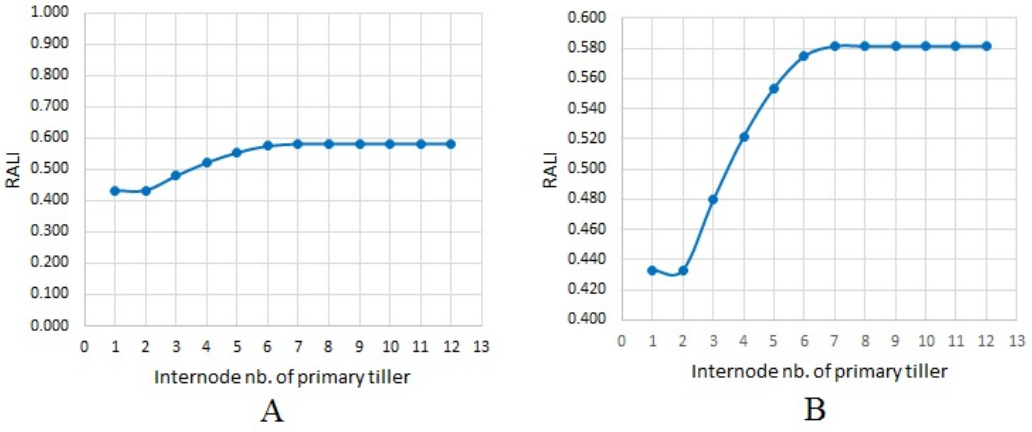

FIGURE 7.16: Graph of the sensitivity analysis result for parameter "internode number of primary tillers". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 9.

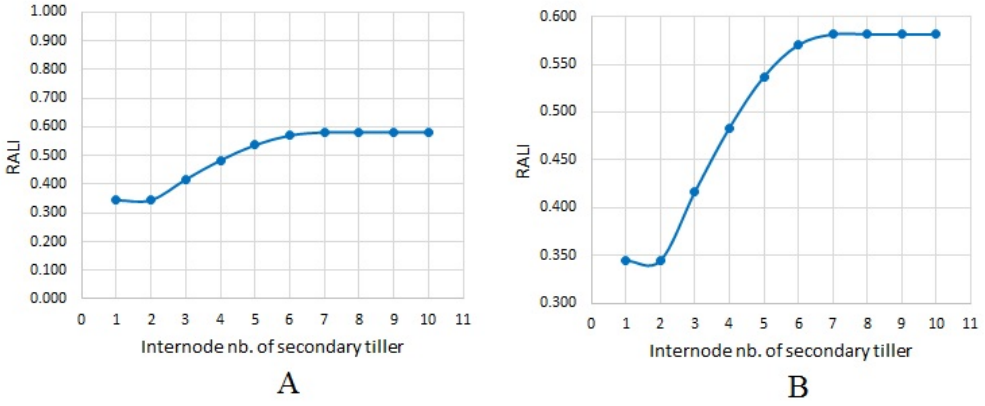

FIGURE 7.17: Graph of the sensitivity analysis result for parameter "internode number of secondary tillers". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 7.

The second sensitivity analysis has been done for the six additional parameters. The first parameter "vegetative days" affects the RALI values growing linearly (see Figure 7.23). This is rational, a plant with a larger number of vegetative days can accumulate more light. In the next graph, the RALI curve grows in the first values, but stays constant then. This happens in the result of the sensitivity analysis for the parameter "internode coefficient" (Figure 7.24). For the next two parameters "final leaf length coefficient" and "leaf length coefficient a1" (Figure 7.25 and 7.26), they give a little effect (small range of values) on the light interception actually, but the trend is increasing with some fluctuations. This phenomenon also occurs in the last two parameters "leaf width coefficient Wa and Wc" (Figure 7.27 and 7.28) analysis, but with a more smooth behaviour. They give a growing effect for RALI, but in a small range of values.

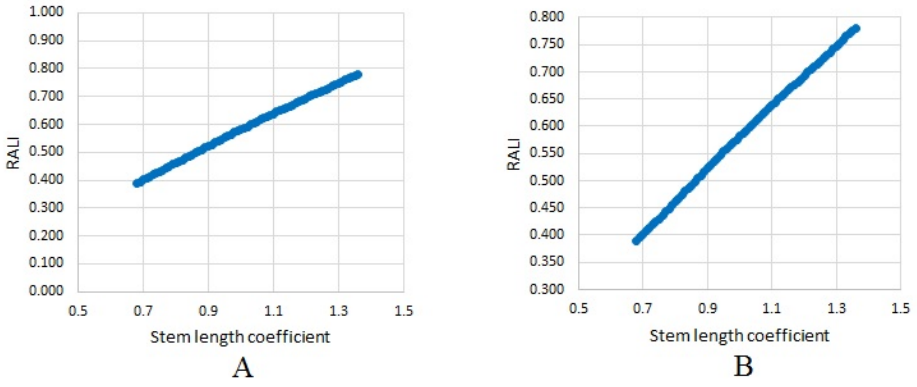

FIGURE 7.18: Graph of the sensitivity analysis result for parameter "stem length coefficient". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 1.0.

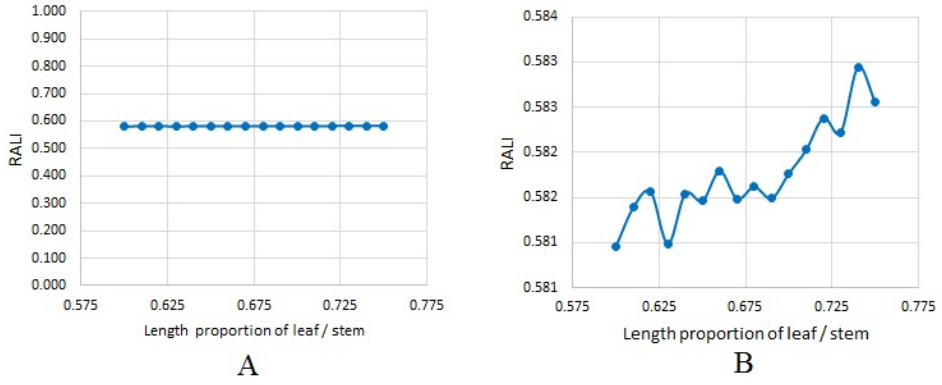

FIGURE 7.19: Graph of the sensitivity analysis result for parameter "length proportion of leaf/stem". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 0.65.

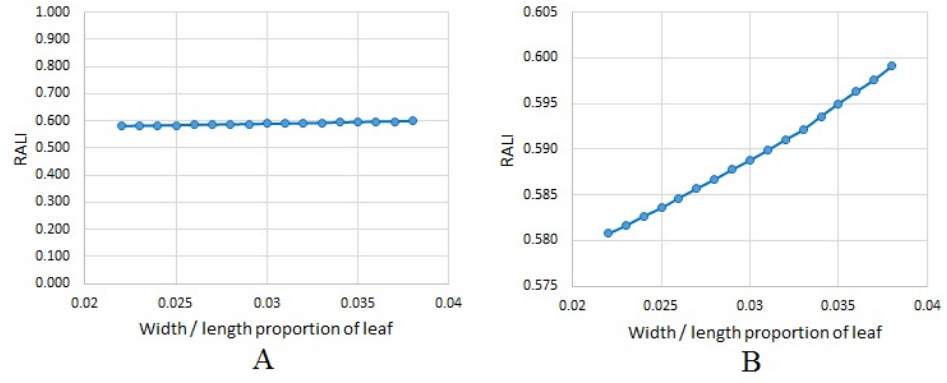

FIGURE 7.20: Graph of the sensitivity analysis result for parameter "width/length proportion of leaf". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 0.022.

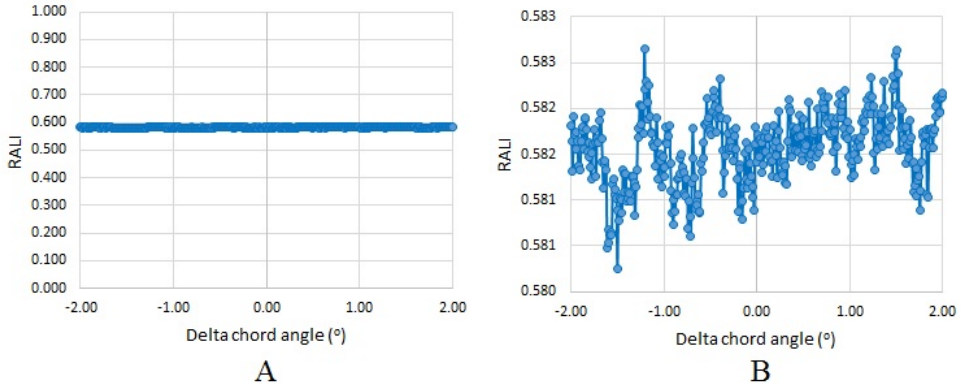

FIGURE 7.21: Graph of the sensitivity analysis result for parameter "delta chord angle". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 0^o.

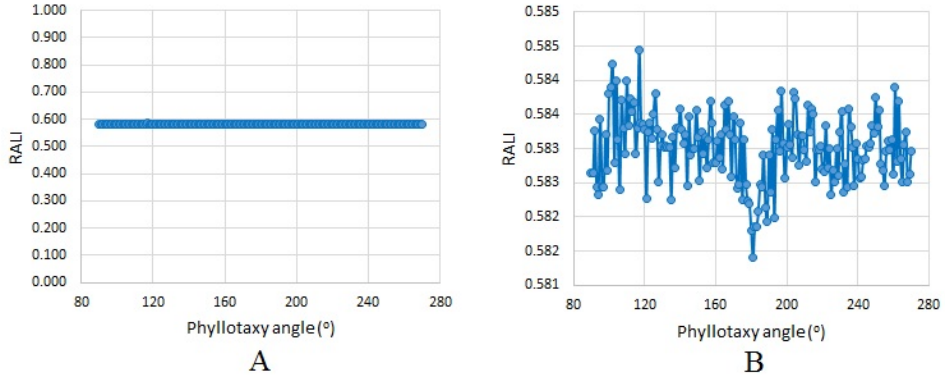

FIGURE 7.22: Graph of the sensitivity analysis result for parameter "phyllotaxy angle". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 180^o.

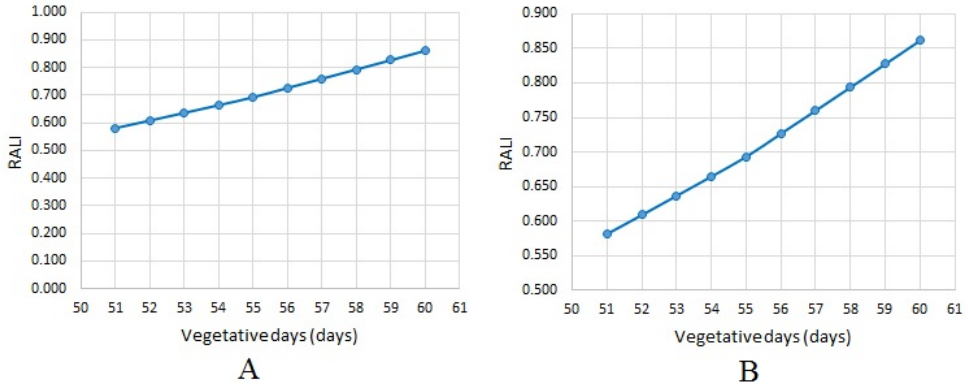

FIGURE 7.23: Graph of the sensitivity analysis result for parameter "vegetative days". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 51 days.

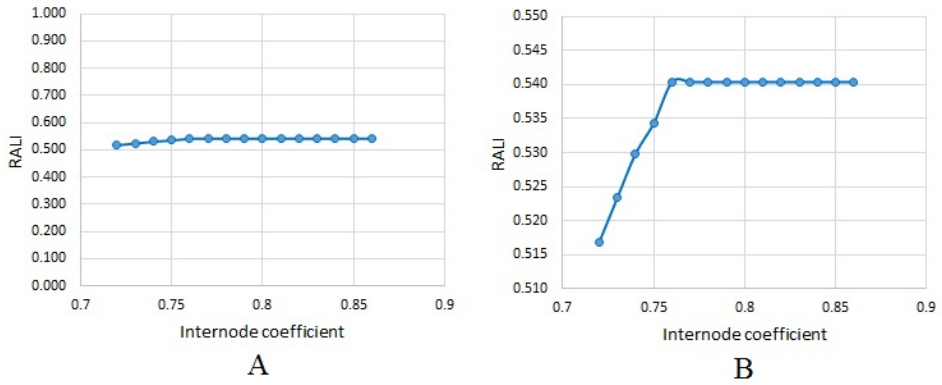

FIGURE 7.24: Graph of the sensitivity analysis result for parameter "internode coefficient". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 0.8286.

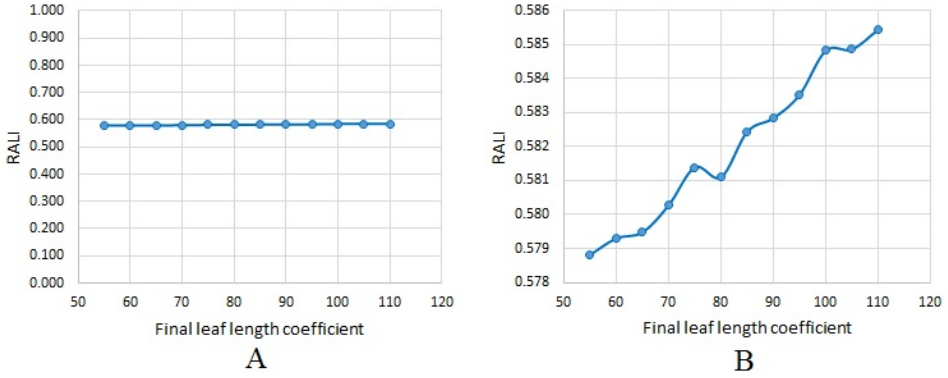

FIGURE 7.25: Graph of the sensitivity analysis result for parameter "final leaf length coefficient". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 78.09.

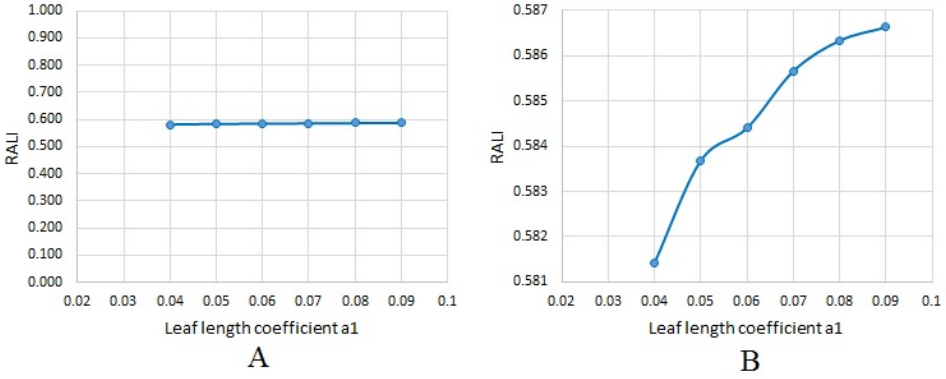

FIGURE 7.26: Graph of the sensitivity analysis result for parameter "leaf length coefficient a1". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is 0.04.

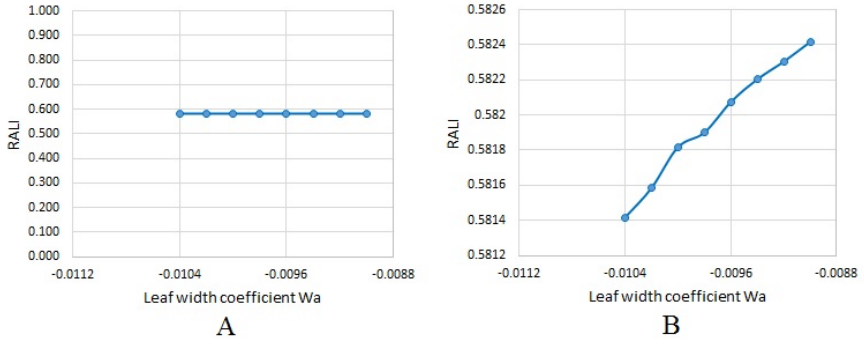

FIGURE 7.27: Graph of the sensitivity analysis result for parameter "leaf width coefficient Wa". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is -0.0104.

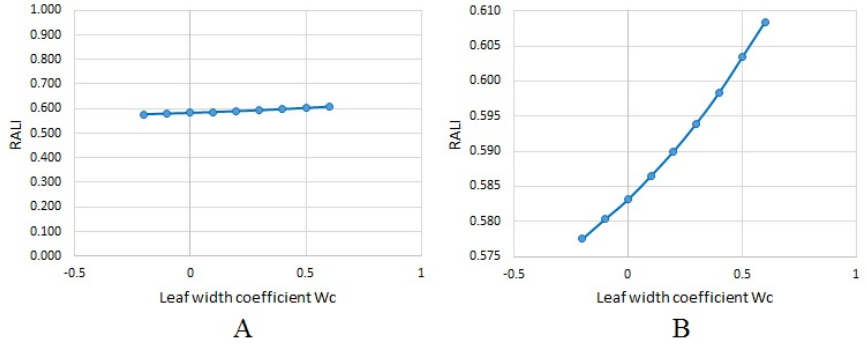

FIGURE 7.28: Graph of the sensitivity analysis result for parameter "leaf width coefficient Wc". (A) The graph view in a complete range. (B) The graph in more detailed view. The assumed value for this parameter of the variety Dodokan is -0.0541.

7.3 Test of the Radiation Model

7.3.1 Experiment on the Number of Rays

An experimental modification of the parameter "number of rays" of our model "skylight" was conducted to see the consistency of the result (the value of light interception) and the runtime of the model. Table 7.1 shows the result of the light interception value (in relative value) through six types of rays number, and the result is graphically delivered by Figure 7.29. The result says that the number of rays 1.00E+06 has the most constant value of light interception. The correction factor (CF) in Table 7.1 was calculated by dividing the maximum mean with the current mean. It is used as a multiplier factor of the result if the current number of rays is chosen to be used. In addition, the experiment result of model runtime can be seen in Figure 7.30.

	Number of rays							
	1.0E+2	1.0E+3	1.0E+4	2.5E+4	5.0E+4	7.5E+4	1.0E+5	1.0E+6
Mean	8.1E-3	1.7E-2	2.6E-1	3.8E-1	7.0E-1	8.3E-1	9.0E-1	9.9E-1
Var.	5.1E-5	1.6E-4	1.5E-3	1.6E-3	1.4E-3	9.9E-4	4.4E-4	2.9E-5
Std.	7.2E-3	1.3E-2	3.9E-2	4.0E-2	3.8E-2	3.1E-2	2.1E-2	5.4E-3
CF.	12	5.7	3.8	2.6	1.4	1.2	1.1	1.0

TABLE 7.1: The experiment result of the effect of the parameter "number of rays" to the light interception. "Var." is the variance, "Std." is the standard deviation, and "CF." is the correction factor.

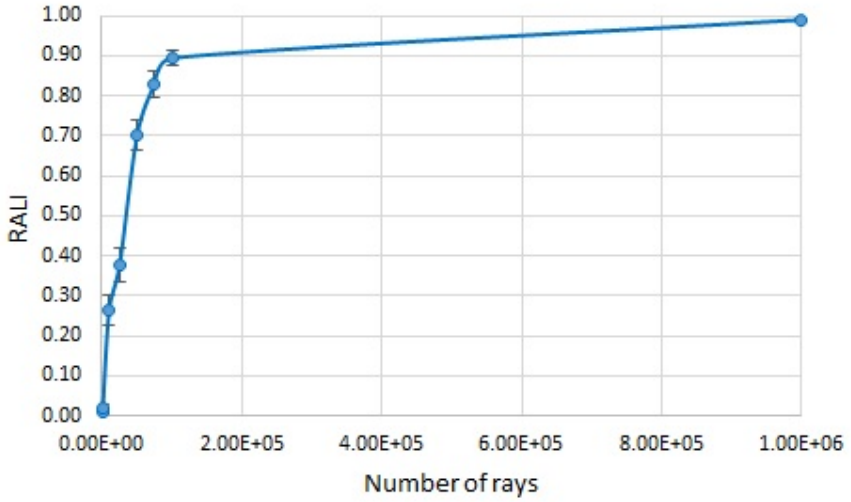

FIGURE 7.29: Graph of the experiment result of the effect of the number of rays to the light interception

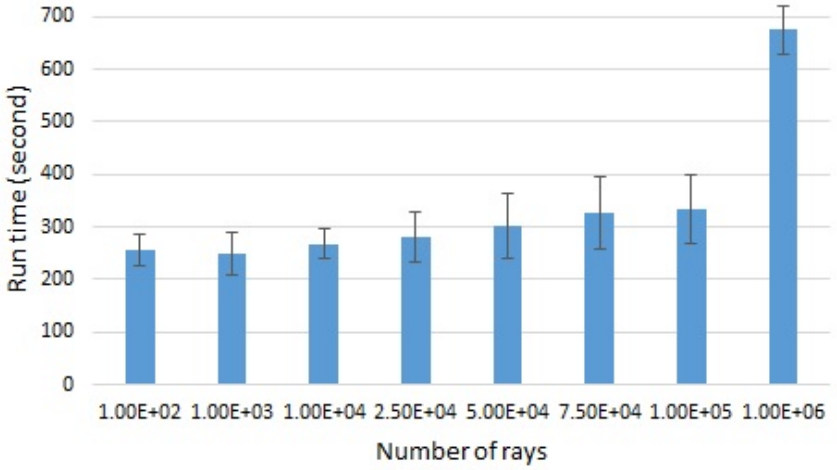

FIGURE 7.30: Graph of the experiment result of the effect of the number of rays to the model runtime

7.3.2 Direct and Diffuse Light

One cluster of our constructed model is the *skylight model*. It can simulate the behaviour of sun and sky based on selected time and geographical position on earth, such as sun movement and power density of direct and diffuse light. Figure 7.31 shows the graph of the sun and sky light power density (in an average relative value) per hour in a day. It is produced by defining the first day in a year for the parameter "day" and -7 for the parameter "latitude", where the latitude -7 is describing the geographical position of Indonesia. The behaviour of sun and sky is similar along the year throughout whole Indonesia. And based on the graph in Figure 7.31, the sun and sky shine (producing the light power) in 13 hours a day, with the highest light power reached at 12 o'clock a.m.

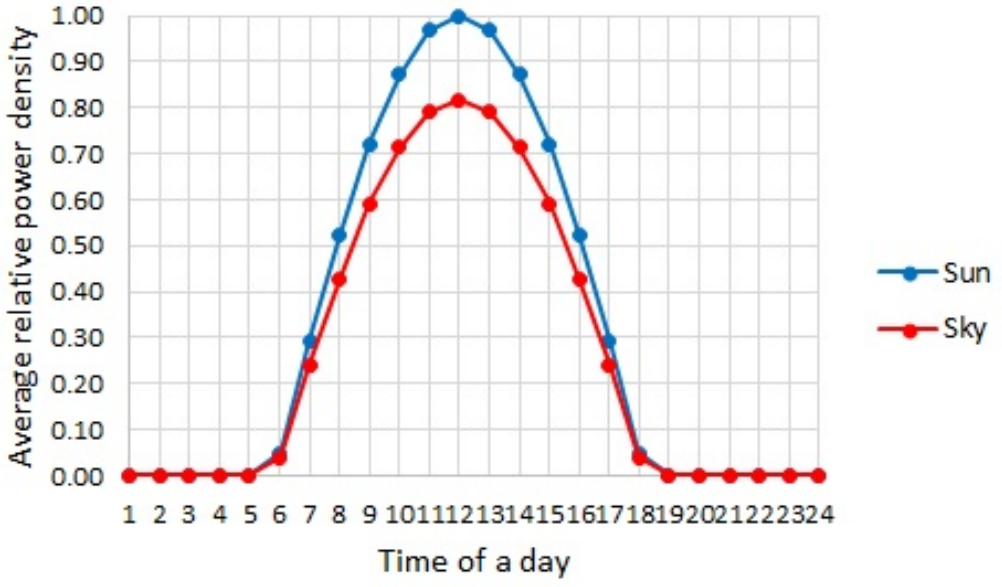

FIGURE 7.31: Graph of sun and sky light power densities, in average relative value per hour in a day

7.4 Optimization Results

Simulated annealing (SA) is one of five optimization methods that we successfully constructed and embedded into our constructed model. Particularly in our research case, based on several previous experiments, SA often showed a good capability to get the optimal value [163]; thus we assumed that SA is the proper method to be used in our main research experiment. Here, finally SA was used as a main method in optimizing a rice plant's capability in intercepting the light. Through two optimization processes starting

from the existing morphologies (of four Indonesian rice varieties), the new morphological shape of the rice plant that optimally absorbs the light was successfully obtained.

By using 15 selected parameters and letting the others fixed to their variety-specific values, the first experiment generated the optimal shape of each rice variety (Figure 7.32). The graph shows that the the variety Inpari 9 has the optimal intercepted light, and each variety's optimal shape has better light interception compared to its natural shape (Figure 7.33). The optimal variety Inpari 9 can reach the optimal light with the parameter combination: starting day to bend = 5, basic leaf chord angle = 68^o, stem diameter = 0.0042m, bending coefficient (p) = 0.44, primary tiller angle = 31^o, secondary tiller angle = 20^o, primary tiller number = 8, secondary tiller number = 9, number of internodes of primary tiller = 10, number of internodes of secondary tiller = 9, stem length coefficient = 1.33, length proportion of leaf/stem = 0.62, width/length proportion of leaf = 0.032, delta chord angle = 1.99^o, and phyllotaxy angle = 134^o.

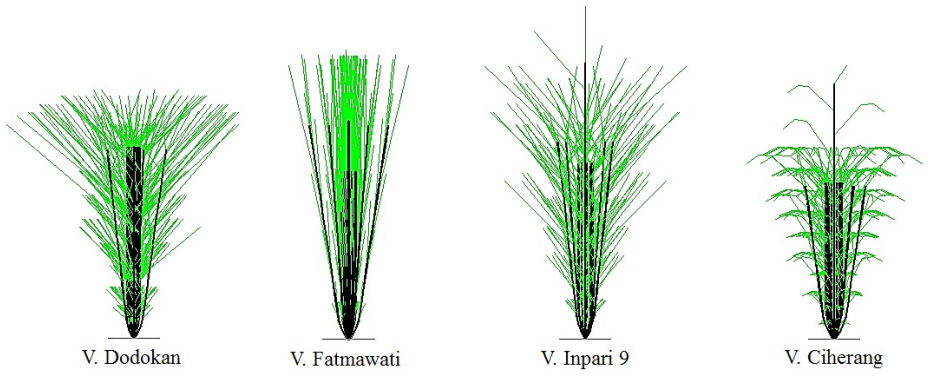

FIGURE 7.32: Optimal shape of the rice plant for each variety

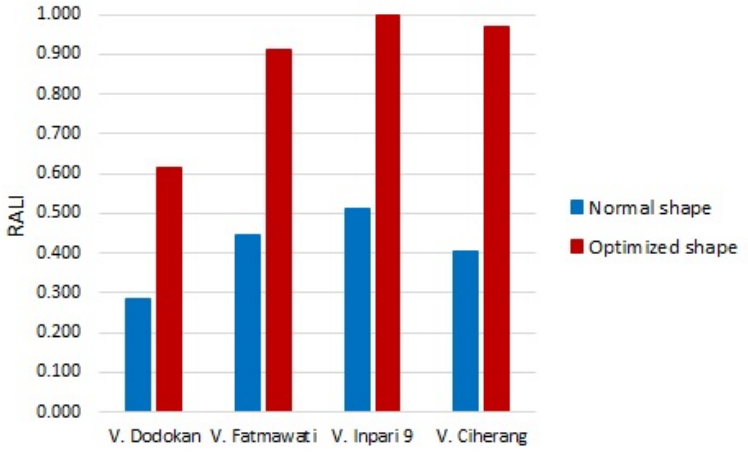

FIGURE 7.33: Graphical comparison of the normal shape and the optimized shape for each variety w.r.t. light interception

In the second optimization, only the six additional parameters were modified. Based on this optimization process, the visual appearance of the plant did not change significantly, but an increase of light interception by 14.59% was achieved (with the combination of six additional parameters that have been achieved: vegetative age = 60 days, internode coefficient = 0.80, final leaf length coefficient = 62.0, leaf length coefficient a1 = 0.07, leaf width coefficient Wa = -0.0097, and leaf width coefficient Wc = 0.3), primarily due to modified leaf dimensions. Further investigations with a higher number of modified parameters will be necessary to confirm these results.

Part IV

Discussion, Conclusion, and Further Works

Chapter 8

Discussion, Conclusion, and Further Works

8.1 Discussion

8.1.1 General Discussion

Talking about the optimization method "simulated annealing" that we used in the virtual experiments, indeed it is a probabilistic method for obtaining the global optimum of an objective function ([15], [26], [82]); however, in high-dimensional, fire-grained parameter space, the global optimum can not be known exactly. In our study, approximately there are more than 1.1×10^{21} and 8.4×10^{5} possibilities of parameter combination respectively in the first and second optimization process. Under this condition, the obtained optimum value of the objective function does not indicate that it is the global optimum. In addition, due to the long model run-time we successfully executed the model only in five repetitions. In one repetition, the model is only able to scan "2 $\times$ number of parameters $\times$ 17" possibilities of parameter combination; here, the value 2 indicates two neighbours of each parameter and the value 17 indicates the number of loop executions for one repetition with the parameters "base temperature" 100^o, "optimum temperature" 40^o, and "adjusting multiplier" 0.95.

Furthermore, because of a lot of possibilities of parameter combinations, and also because of the technical characteristics belonging to the method "simulated annealing", the process of optimal value seeking can get stuck in one area of local optimum with a parameter combination describing a specific characteristic of the rice plant structure. There is the possibility that the "proposed" optimal shape of the rice plant has a different shape compared with the "natural" shape; although, we have technically defined all

involved parameters based on references and empirical data ([13], [69], [72], [105], [146], [155], [183], [189]).

Regarding another important issue of the model, the mechanism of leaf bending, we proposed a simple leaf bending model. We used several geometrical parameters such as leaf chord angle, leaf length, and leaf tip position (please see Figures 6.5 and 6.6). The direction and quantity of leaf bending depend on the bending coefficient (p), where it can be calculated by dividing parameter H with a half of leaf length. This part of our study is similar with [187]. They evaluated the degree of rice leaf bending by using p-type Fourier descriptors, specifically for the rice sprout leaf bending. However, what [116] conducted is different. [116] used the factor leaf-weight for realizing a bending mechanism. They also implemented it as part of the general morphology of the maize leaf.

Based on the virtual experiment of our study, three from four optimized varieties have erect leaves (see Figure 7.32). This result is in line with the statement from [104]. They said that one of commonly required leaf characteristics in contemporary rice breeding is erect form. However [143] concluded differently, in their research they concluded that the leaves of an erect canopy receive lower irradiance than the leaves of a prostrate canopy.

In addition, to produce the erect leaf, it should be smaller and shorter [104]; however our experiment result said that the optimal plant has longer and wider leaves. The values 0.62 and 0.032 for parameters "length proportion of leaf/stem" and "width/length proportion of leaf" respectively indicate that the leaves are longer and wider. By using equation (6.29), the current leaf length is 0.67m, it means that the optimized leaf length is 34.96% longer than the natural maximum leaf length (0.49m). In addition, by using equation (6.31), the current leaf width is 0.021m, it means that the optimized leaf width is 64.11% wider than the natural maximum leaf width (0.013m).

8.1.2 Comparison with Other Study

Improving the capacity of photosynthesis is envisaged as one of the approaches to increase the productivity of crop plants [100] [188]. The challenge how to identify a new ideal type of canopy of a plant that can get a higher capacity in photosynthesis is one thing to study deeper. To do so, [149] developed a model of a canopy by considering three components: a canopy architectural model, a forward ray tracing algorithm, and a steady-state biochemical model of photosynthesis. This study used the variety of Indica rice Teqing that has been planted in Beijing in June 2009 (in 150 days) as a research object. The parameters considered to define the plant structure are the number of tillers,

the number of leaves, leaf base height, leaf length, leaf width, leaf angle, and leaf curvature. The simple view of a single rice tiller (a), leaf (b), and leaf length and width (c) from this study is shown in Figure 8.1. Also, the single 3-d plant (a) and several plants (b) with 14 tillers are displayed in Figure 8.2.

Practically, they developed a model for simulating the light distribution in a canopy by using their canopy architecture model as an input. Three types of light (direct, diffuse and scattered) are simulated here. The value of photosynthetic photon flux density (PPFD) can be predicted by using the forward ray tracing algorithm. Especially for transmitted and reflected light rays, they are technically traced by using a Monte Carlo approach similar to ones.

As part of the result of this paper, a framework for designing an ideal crop architecture in gaining an optimal canopy CO_2 uptake rate (Ac) under climate condition changes was proposed. Manipulations of stem height, leaf width, and leaf angle can affect the canopy photosynthesis. By combining the detailed light environment in a canopy with a steady-state biochemical model of C3 photosynthesis, the daily total canopy CO_2 uptake rate can be simulated.

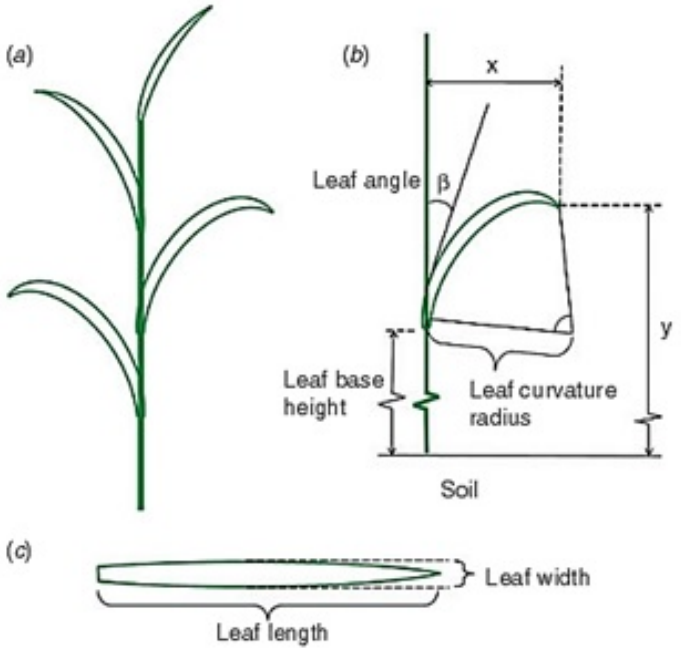

FIGURE 8.1: Simple view of single rice tiller (a), rice leaf (b), and leaf length and width (c) [149]

Essentially, several things what [149] conducted are similar to what we did, such as the research objective which is to find the ideal type of canopy and the usage of a Monte

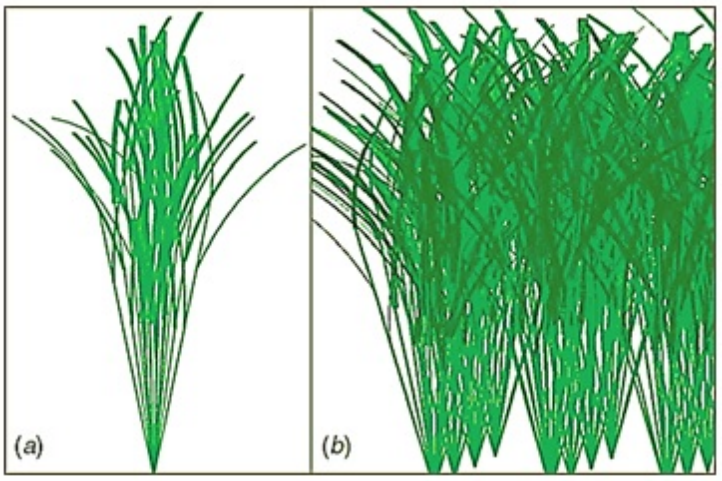

FIGURE 8.2: A simple 3-d single rice plant (a) and several rice plants (b) with 14 tillers [149]

Carlo approach in ray tracing. However, what we achieved is dissimilar in several parts. They calculated the specific value of canopy CO_2 uptake rate to find the optimal rice canopy; on the other hand, we use only the light interception calculation. However, we not only calculated the light intercepted by leaves, but also by stem organs.

As research object, they used only one type of rice variety in doing virtual experiments, but they conducted some experiments in several plants; while we used four rice varieties, although we only conducted the virtual experiments in a single plant of each variety. Moreover, to see the influence of canopy photosynthesis, [149] changed three parameters in their sensitivity analysis with three or six fixed values of each parameter; in contrast, we used fifteen selected architectural parameters. Furthermore, we recommended an optimal shape of the plant by interpolating the rice plant shape in each variety and also among varieties (to propose a "new variety" plant shape). Due to the large number of possibilities in finding the optimum shape, we applied five types of optimization models to approximate global optima.

Regarding the sensitivity analysis, we used this analysis before we interpolated the new optimum shape of the plant. We did it by using one (variety Dodokan) of four rice varieties we studied. [149] did this analysis to see the influence of the changes of several parameters on canopy photosynthesis (Ac becomes the analysis indicator), they conducted their analysis in three or six different combinations of CO_2 and temperature. On the other hand, we did it to see the effect of the value changes of fifteen parameters on light interception, where the relative accumulated light interception (RALI) becomes the analysis indicator. CO_2 and temperature regime were not changed in our study.

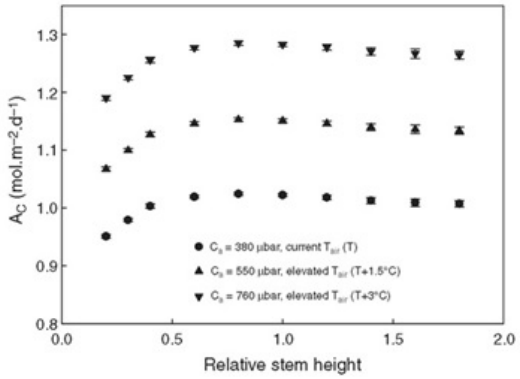

FIGURE 8.3: Predicted influence of stem length on canopy photosynthesis in three different combinations of CO_2 and temperature [149]

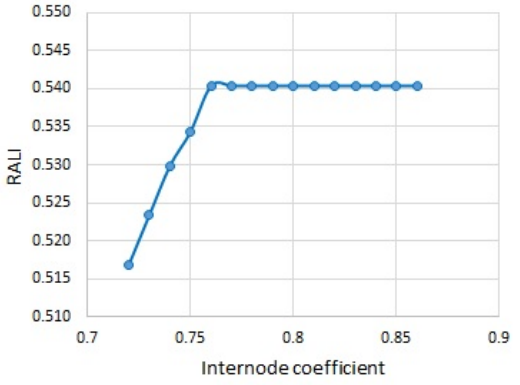

FIGURE 8.4: The effect of internode coefficient on relative accumulated light interception (RALI), from our study. The fitted value for this parameter of the variety Dodokan (our research object used in the analysis) is 0.8286.

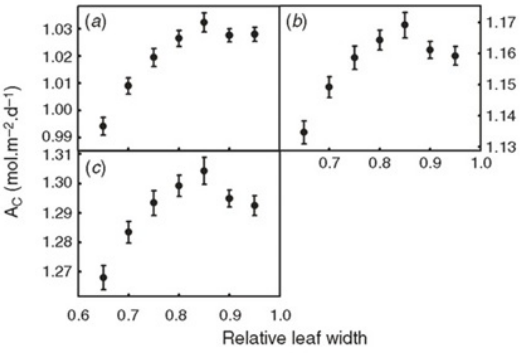

FIGURE 8.5: Predicted influence of leaf width on canopy photosynthesis in three different combinations of CO_2 and temperature [149]. "Relative leaf width" is a factor to be multiplied with the current leaf width.

For the parameter "stem length", [149] got the result of a curve pattern like in Figure 8.3. They conducted it in three different combinations of CO_2 and temperature. The result shows that the value of Ac increases in the first stem lengths, and remains constant then. A similar pattern occurs with the effect of "internode coefficient" changes on light interception that we obtained, it is shown by Figure 8.4. The result was that the values of RALI grow linearly in the first values and stay constant then. In addition, with the parameter "leaf width", based on the sensitivity analysis, [149] got a curve pattern like in Figure 8.5; the values of Ac increase slowly, but go down for the last several values. Oppositely, we conducted a similar sensitivity analysis of the "proportion width / length of leaf", and the result (Figure 8.6) displays that the values of RALI increase linearly. Finally, for the sensitivity analysis of "leaf angle", the [149] result of the first three experiments (from a canopy with low LAI) presents that Ac slowly increases (from a canopy with high LAI); but in the next other three experiments, the pattern of the Ac curve is relatively fluctuating (Figure 8.7). Based on our analysis, the curve pattern of the effect of "leaf chord angle" to light interception is fluctuating with an increasing trend (Figure 8.8), similar to the pattern in the first three experiments from [149]. [149] obtained their results (Figure 8.3, 8.5, 8.7) from a virtual canopy consisting of 64 rice plants, whereas our results came from a single virtual rice plant (Figure 8.4, 8.6, 8.8).

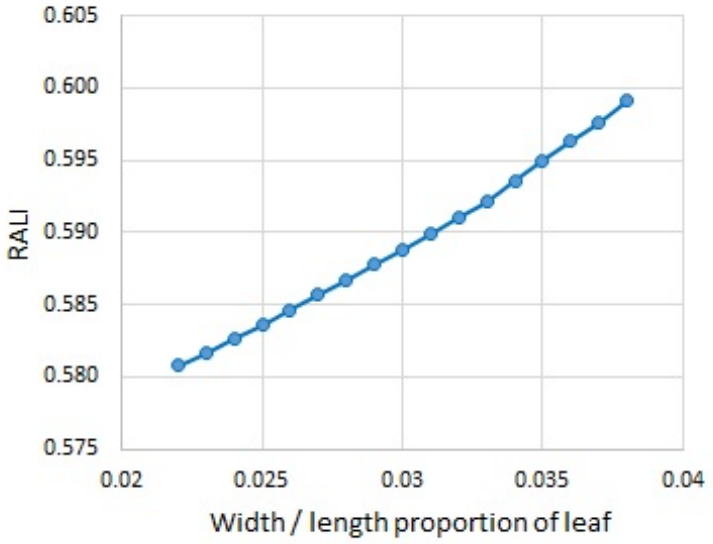

FIGURE 8.6: The effect of width/length proportion of leaf on relative accumulated light interception (RALI), from our study. The fitted value for this parameter of the variety Dodokan (our research object used in the analysis) is 0.022.

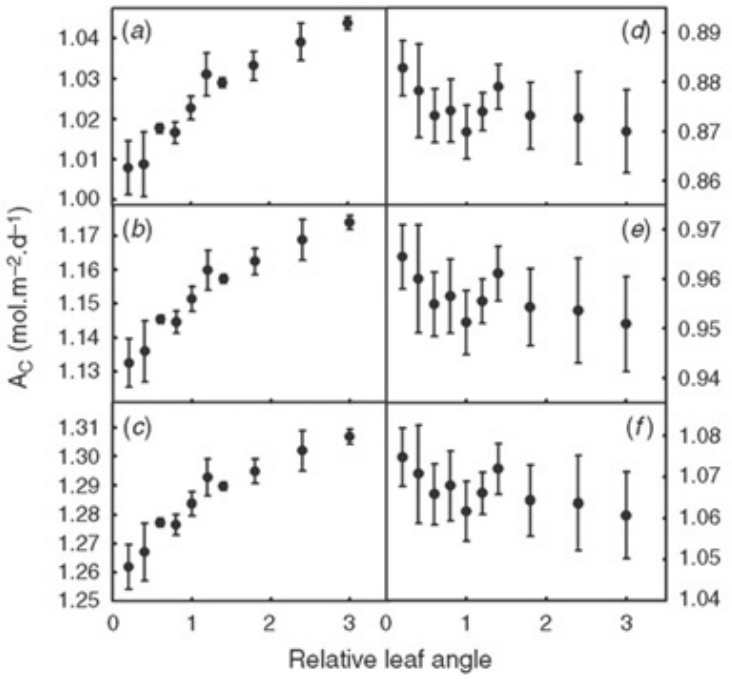

FIGURE 8.7: Predicted influence of leaf angle on canopy photosynthesis in six different combinations of CO_2 and temperature [149]. The leaf angle used by [149] is not the chord angle, but the angle under which the leaf emerges from the main stem.

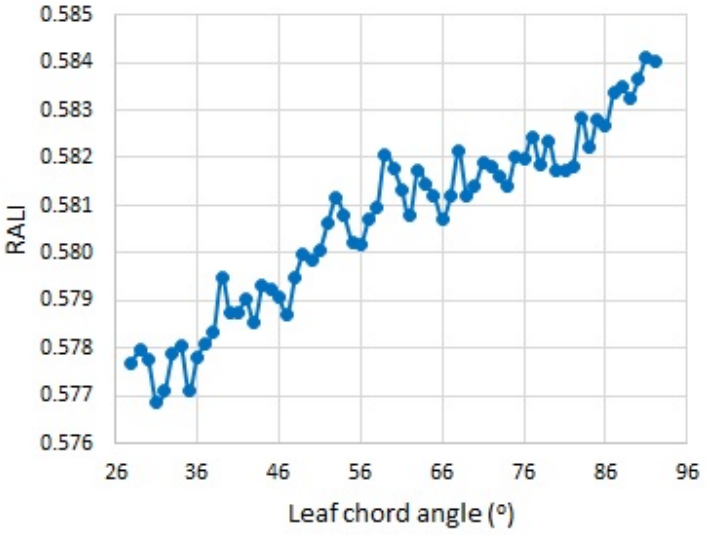

FIGURE 8.8: The effect of leaf chord angle on relative accumulated light interception (RALI), from our study. The fitted value for this parameter of the variety Dodokan (our research object used in the analysis) is 70^o.

8.2 Conclusion

Functional-structural plant modelling and GroIMP are respectively a methodology and modelling platform software that we used in constructing our model. There are three clusters of the model that we have successfully constructed. Those model clusters are the rice plant, skylight, and optimization. The cluster "rice plant" simulates the morphology, growth, and development of an individual above-ground rice plant. The vegetative organs (leaves, stem, and tillers) are the green organs that are modeled through this model cluster. Practically, the first model cluster implements the functional-structural model of four Indonesian rice varieties based on empirical data.

Furthermore, based on the Indonesian geographical condition, the model cluster "skylight" models the sun movement and intensity of both direct and diffuse light. The interconnection between the model clusters rice plant and skylight enables the model to calculate the amount of light that can be intercepted by green organs of the plant, especially during the vegetative phase (here, the amount of the intercepted light, put in relation to its maximum, is called relative accumulated light interception or RALI). And then, the third model cluster "optimization" that consists of five types of optimization model (full factorial design, simple random sampling, Latin hypercube sampling, hill climbing and simulated annealing) is able to optimize the 3-d structure of the rice plant in intercepting the light. Here, the light intercepted by green organs of an individual rice plant during the vegetative phase is the objective function of the optimization.

Fifteen structural parameters of the plant are involved in our first optimization approach. The sensitivity analysis of those parameters can show the gradual effects on light interception based on stepwise small changes of parameter values. By using the optimization model "simulated annealing" and through interpolating the empirical values of parameters, the predicted optimal morphological shape of the plant for each variety can be rationally produced. The optimization model "simulated annealing" was chosen as an appropriate model to implement, because it has obtained the global optimum many times in several more simple experiments.

The next six parameters that describe the general plant structure are also treated by using sensitivity analysis. And a second optimization and interpolating process has been conducted as well. Based on this second optimization and interpolating processes, the new optimal shape of a rice plant can be proposed. This shape does not only describe the new form of plant, but also proposes a new combination of values of parameters that is not related to one single specific variety.

The recommended new optimal shape of the rice plant is characterized by the first 15 structural parameters and the second 6 parameters. Those values of all parameters are delivered below:

1. Starting day to bend = 5
2. Leaf chord angle = 68^o
3. Stem diameter = 0.0042m
4. Bending coefficient (p) = 0.44
5. Primary tiller angle = 31^o
6. Secondary tiller angle = 20^o
7. Number of first-order tillers = 8
8. Number of second-order tillers = 9
9. Internode number of first-order tiller = 10
10. Internode number of second-order tiller = 9
11. Stem length coefficient = 1.33
12. Length proportion of leaf / stem = 0.62
13. Width / length proportion of leaf = 0.032
14. Delta chord angle = 1.99^o
15. Phyllotaxy angle = 134^o
16. Vegetative days = 60 days
17. Internode coefficient = 0.84
18. Final length coefficient = 62.0
19. Leaf length coefficient a1 = 0.07
20. Leaf width coefficient Wa = -0.0097
21. Leaf width coefficient Wc = 0.3

Based on the final experiment, the optimized shape of four Indonesian rice varieties can reach RALI value 0.875 (on average), it is much higher than the RALI average value of four Indonesian varieties in normal shape (0.412). This might suggest that there is a big opportunity to increase the capability of Indonesian rice plants in intercepting light.

8.3 Further Works

Indeed, parameters representing nitrogen and water content have been defined in the model. Specifically in the model cluster "rice plant", two parameters "leaf-water content" and "leaf-nitrogen content" were defined as floating point parameters; where they respectively mean the water and nitrogen content of the leaf. These parameters will possibly be involved in the optimization process in further work.

Moreover, in this study we conducted the sensitivity analysis for each single parameter we used in the optimization process. The sensitivity analysis for the combination of two or three parameters should be done in further work. The analysis result will be able to deliver more fruitful information, because interaction effects of several parameters will then be taken into account.

Regarding the experiment, more repetitions of the optimization process can increase the possibility to reach the global optimum value of light interception; although, the run time of the model will be a specific challenge in the optimization experiments. In this research, we only conducted five and two repetitions for the first and second optimization processes respectively.

Defining other objective functions also can be done to extend the study, such as the yield optimization for a single or several rice plants. In the research field in plant breeding as well, the aspect of the biological realizability of the theoretically-optimal shape of a rice plant is one interesting further work to do. Possibly some parameters which we used in our study can not be changed independently of each other because of genetical or biophysical reasons.

Regarding the leaf bending mechanism, as has been mentioned in the part Discussion, the leaf-weight factor that is tightly related with gravity is an important parameter to be considered. The inclusion of a proper, physically-based model of plant biomechanics would be a promising next step to enable the involvement of the leaf-weight factor and to get a more realistic representation of the trade-off between light-interception capacity and the costs for mechanical support of the plant organs.

Bibliography

[1] Abe, A., Kosugi, S., Yoshida, K., Natsume, S., Takagi, H., Kanzaki, H., Matsumura, H., Yoshida, K., Mitsuoka, C., Tamiru, M., et al. Genome sequencing reveals agronomically important loci in rice using mutmap. *Nature biotechnology 30*, 2 (2012), 174–178.

[2] Abelson, H., and DiSessa, A. *Turtle geometry: The computer as a medium for exploring mathematics*. MIT press, 1986.

[3] Agrawal, A. A., Laforsch, C., and Tollrian, R. Transgenerational induction of defences in animals and plants. *Nature 401*, 6748 (1999), 60–63.

[4] Ahmed, S. Methods in sample surveys. *Johns Hopkins Bloomberg School of Public* (2009).

[5] Aistleitner, C., Hofer, M., and Tichy, R. A central limit theorem for latin hypercube sampling with dependence and application to exotic basket option pricing. *International Journal of Theoretical and Applied Finance 15*, 07 (2012), 1250046.

[6] Anderson, M. C. Stand structure and light penetration. II. A theoretical analysis. *Journal of Applied Ecology* (1966), 41–54.

[7] Ashlock, D. *Evolutionary computation for modeling and optimization*. Springer Science & Business Media, 2006.

[8] Asner, G. P., Scurlock, J. M., and A Hicke, J. Global synthesis of leaf area index observations: implications for ecological and remote sensing studies. *Global Ecology and Biogeography 12*, 3 (2003), 191–205.

[9] Barczi, J.-F., Rey, H., Caraglio, Y., De Reffye, P., Barthélémy, D., Fourcaud, T., and Dong, Q. X. Amapssim: an integrative whole-plant architecture simulator based on botanical knowledge. *Annals of Botany 101*, 8 (2008), 1125–1138.

[10] Bardenas, E. A., and Chang, T.-T. *Morphology and Varietal Characteristics of the Rice Plant*. Int. Rice Res. Inst., 1965.

[11] Barnabás, B., Jäger, K., and Fehér, A. The effect of drought and heat stress on reproductive processes in cereals. *Plant, Cell & Environment 31*, 1 (2008), 11–38.

[12] Barnes, J. *Complete Works of Aristotle, The Revised Oxford Translation*, vol. 1. Princeton University Press, 1984.

[13] Bartlett, M. E., and Thompson, B. Meristem identity and phyllotaxis in inflorescence development. *Frontiers in plant science 508*, 5 (2014), 1–11.

[14] Bastiaans, L., Kropff, M., Kempuchetty, N., Rajan, A., and Migo, T. Can simulation models help design rice cultivars that are more competitive against weeds? *Field Crops Research 51*, 1 (1997), 101–111.

[15] Bertsimas, D., Tsitsiklis, J., et al. Simulated annealing. *Statistical science 8*, 1 (1993), 10–15.

[16] Bolshakova, E. Programming paradigms in computer science education.

[17] Booch, G., Maksimchuk, R. A., Engle, M. W., Young, B. J., Conallen, J., and Houston, K. A. *Object-Oriented Analysis and Design with Applications (3rd Edition)*. Addison-Wesley, 2007.

[18] Bouman, B., Van Keulen, H., Van Laar, H., and Rabbinge, R. The 'school of de Wit' crop growth simulation models: a pedigree and historical overview. *Agricultural Systems 52*, 2 (1996), 171–198.

[19] Boysen-Jensen, P., et al. *Stoffproduktion der Pflanzen*. Fischer, 1932.

[20] Bronstein, A. M., Bronstein, M. M., and Kimmel, R. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[21] Bryan, K. *Quasi-Newton Methods*. Rose-Hulman Institute of Technology, 2004.

[22] Buck-Sorlin, G. *The Period of Rice Plant Growth*. http://www.knowledgebank.grogra.de, accessed: 25 July 2015, 2010.

[23] Buck-Sorlin, G., Hemmerling, R., Kniemeyer, O., Burema, B., and Kurth, W. A rule-based model of barley morphogenesis, with special respect to shading and gibberellic acid signal transduction. *Annals of Botany 101*, 8 (2008), 1109–1123.

[24] Cao, H.-x., Yan, L., Liu, Y.-x., Hanan, J. S., Yue, Y.-b., Zhu, D.-w., Lu, J.-f., Sun, J.-y., Shi, C.-l., Ge, D.-k., et al. Biomass-based rice (Oryza sativa L.) aboveground architectural parameter models. *Journal of Integrative Agriculture 11*, 10 (2012), 1621–1632.

[25] Cellier, P., Ruget, F., Chartier, M., and Bonhomme, R. Estimating the temperature of a maize apex during early growth stages. *Agricultural and Forest Meteorology 63*, 1 (1993), 35–54.

[26] Černỳ, V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications 45*, 1 (1985), 41–51.

[27] Chang, T.-T. The origin, evolution, cultivation, dissemination, and diversification of Asian and African rices. *Euphytica 25*, 1 (1976), 425–441.

[28] Cheng, J., and Druzdzel, M. J. Latin hypercube sampling in Bayesian networks. In *FLAIRS Conference* (2000), pp. 287–292.

[29] Chomsky, N. Three models for the description of language. *IRE Transactions on Information Theory 2*, 3 (1956), 113–124.

[30] Costes, E., and Guédon, Y. Modelling branching patterns on 1-year-old trunks of six apple cultivars. *Annals of Botany 89*, 5 (2002), 513–524.

[31] Costes, E., Smith, C., Renton, M., Guédon, Y., Prusinkiewicz, P., and Godin, C. MAppleT: simulation of apple tree development using mixed stochastic and biomechanical models. *Functional Plant Biology 35*, 10 (2008), 936–950.

[32] Dauzat, J., Clouvel, P., Luquet, D., and Martin, P. Using virtual plants to analyse the light-foraging efficiency of a low-density cotton crop. *Annals of Botany 101*, 8 (2008), 1153–1166.

[33] Dauzat, J., and Eroy, M. Simulating light regime and intercrop yields in coconut based farming systems. *European Journal of Agronomy 7*, 1-3 (1997), 63–74.

[34] Dauzat, J., Rapidel, B., and Berger, A. Simulation of leaf transpiration and sap flow in virtual plants: model description and application to a coffee plantation in Costa Rica. *Agricultural and Forest Meteorology 109*, 2 (2001), 143–160.

[35] Davey, K. R. Latin hypercube sampling and pattern search in magnetic field optimization problems. *Magnetics, IEEE Transactions on 44*, 6 (2008), 974–977.

[36] Dawkins, P. *Newton Method*. Paul Online Math Notes, 2013.

[37] De Datta, S. K. *Principles and practices of rice production.* Int. Rice Res. Inst., 1981.

[38] De Reffye, P., Houllier, F., Blaise, F., Barthelemy, D., Dauzat, J., and Auclair, D. A model simulating above-and below-ground tree architecture with agroforestry applications. *Agroforestry Systems 30*, 1-2 (1995), 175–197.

[39] Deckmyn, G., Evans, S. P., and Randle, T. J. Refined pipe theory for mechanistic modeling of wood development. *Tree Physiology 26*, 6 (2006), 703–717.

[40] Deutsch, J. L., and Deutsch, C. V. Latin hypercube sampling with multidimensional uniformity. *Journal of Statistical Planning and Inference 142*, 3 (2012), 763–772.

[41] Dogniaux, R. Exposition: énergétique par ciel serein des parois orientées et inclinées. données d'application pour la belgique. *Inst. Météo. Belgique. Ed: Misc., séries B*, 25: 153 pp., 26: 86 pp.

[42] Espana, M. L., Baret, F., Aries, F., Chelle, M., Andrieu, B., and Prévot, L. Modeling maize canopy 3d architecture: Application to reflectance simulation. *Ecological Modelling 122*, 1 (1999), 25–43.

[43] Everitt, B. S. *Introduction to optimization methods and their application in statistics.* Springer Science & Business Media, 1987.

[44] Feng, L., Mailhol, J.-C., Rey, H., Griffon, S., Auclair, D., and De Reffye, P. Comparing an empirical crop model with a functional structural plant model to account for individual variability. *European Journal of Agronomy 53* (2014), 16–27.

[45] Fine, S., Singer, Y., and Tishby, N. The hierarchical hidden Markov model: Analysis and applications. *Machine Learning 32*, 1 (1998), 41–62.

[46] Fisher, R. A. On the random sequence. *Quarterly Journal of the Royal Meteorological Society 52*, 250 (1926).

[47] Fogel, D. B. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks 5*, 1 (1994), 3–14.

[48] Foulds, L. R. *Optimization techniques: an introduction.* Springer Science & Business Media, 2012.

[49] Fournier, C., and Andrieu, B. Adel-maize: an L-system based model for the integration of growth processes from the organ to the canopy. Application

to regulation of morphogenesis by light availability. *Agronomie 19*, 3-4 (1999), 313–327.

[50] Fournier, C., Andrieu, B., Ljutovac, S., and Saint-Jean, S. Adel-wheat: a 3d architectural model of wheat development. *Plant Growth Modeling and Applications (eds B.-G. Hu & M. Jaeger)* (2003), 54–66.

[51] Fournier, M. *Mécanique de l'arbre sur pied: maturation, poids proper, constraints climatiques dans la tige standard*. PhD thesis, INP de Lorraine, 1989.

[52] Frazer, G. W., Canham, C., and Lertzman, K. Gap light analyzer (gla), version 2.0: Imaging software to extract canopy structure and gap light transmission indices from true-colour fisheye photographs, users manual and program documentation. *Simon Fraser University, Burnaby, British Columbia, and the Institute of Ecosystem Studies, Millbrook, New York 36* (1999).

[53] Frerichs, R. R. *Rapid Surveys*. unpublished, http://www.ph.ucla.edu/epi/rapidsurveys/RScourse, accessed: 20 July 2015, 2008.

[54] Gao, L., Jin, Z., Huang, Y., and Zhang, L. Rice clock model—a computer model to simulate rice development. *Agricultural and Forest Meteorology 60*, 1 (1992), 1–16.

[55] Goudriaan, J., and Van Laar, H. Modelling potential crop growth processes: textbook with exercises. Tech. rep., Kluwer academic publishers, 1994.

[56] GROGRA. *NURBSSurface code for single leaf*. http://www.grogra.de, accessed: 17 September 2015, 2007.

[57] Guédon, Y., Barthélémy, D., Caraglio, Y., and Costes, E. Pattern analysis in branching and axillary flowering sequences. *Journal of Theoretical Biology 212*, 4 (2001), 481–520.

[58] Guimaraes, E. P. Rice breeding. In *Cereals*. Springer, 2009, pp. 99–126.

[59] Guo, Y., Ma, Y., Zhan, Z., Li, B., Dingkuhn, M., Luquet, D., and De Reffye, P. Parameter optimization and field validation of the functional–structural model Greenlab for maize. *Annals of Botany 97*, 2 (2006), 217–230.

[60] Han, L., Costes, E., Boudon, F., Cokelaer, T., Pradal, C., Da Silva, D., and Faivre, R. Investigating the influence of geometrical traits on light interception efficiency of apple trees: a modelling study with MAppleT. In *Plant Growth Modeling, Simulation, Visualization and Applications (PMA), 2012 IEEE Fourth International Symposium on* (2012), IEEE, pp. 152–159.

[61] Hanada, K. *Chapter 4 – Tillers. In Science of the Rice Plant.* Tokyo: Food and Agriculture Policy Research, 1993.

[62] Hay, R. Harvest index: a review of its use in plant breeding and crop physiology. *Annals of Applied Biology 126*, 1 (1995), 197–216.

[63] Helton, J. C., and Davis, F. J. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliability Engineering & System Safety 81*, 1 (2003), 23–69.

[64] Hemmerling, R., Kniemeyer, O., Lanwert, D., Kurth, W., and Buck-Sorlin, G. The rule-based language XL and the modelling environment GroIMP illustrated with simulated tree competition. *Functional Plant Biology 35*, 10 (2008), 739–750.

[65] Henke, M. Groimp v1.4.2 introduction and overview. International Summer School 'Modelling Ecosystems by Tools from Computer Science. Prague, 16 September, 2013.

[66] Hopkins, W. G., and Hüner, N. P. *Introduction to plant physiology*, vol. 355. Wiley New York, 1995.

[67] howstuffworks.com. *Rice plant parts.* http://science.howstuffworks.com/life/botany/rice-info1.htm, accessed: 26 July 2015, 2008.

[68] Hua, J., and M., K. Optimize tree shape: targeting for best light interception. In *Proceeding of the 7th International Conference on Functional-Structural Plant Models, 2013* (2013).

[69] IAARD. *Buku Panduan Sistem Karakterisasi dan Evaluasi Tanaman Padi.* Indonesian Agency for Agricultural research and Development, Ministry of Agriculture, Jakarta, 2003.

[70] IAARD. *Peningkatan Produksi Padi Melalui Pelaksanaan IP Padi 400.* Indonesian Agency for Agricultural research and Development, Ministry of Agriculture, Jakarta, 2009.

[71] IRRI. *The Period of Rice Plant Growth.* International Rice Research Institute (IRRI), http://www.knowledgebank.irri.org/decision-tools/growth-stages-and-important-management-factors, accessed: 19 July 2015, 2015.

[72] Itoh, J.-i., Hibara, K.-i., Kojima, M., Sakakibara, H., and Nagato, Y. Rice decussate controls phyllotaxy by affecting the cytokinin signaling pathway. *The Plant Journal 72*, 6 (2012), 869–881.

[73] Jacob, C. Genetic L-system programming. In *Parallel Problem Solving from Nature—PPSN III*. Springer, 1994, pp. 333–343.

[74] Jacob, C. Genetic L-system programming: Breeding and evolving artificial flowers with Mathematica. In *Proceedings of the First International Mathematica Symposium* (1995), pp. 215–222.

[75] Jaffuel, S., and Dauzat, J. Synchronism of leaf and tiller emergence relative to position and to main stem development stage in a rice cultivar. *Annals of Botany 95*, 3 (2005), 401–412.

[76] Jennings, P. R., and Aquino, R. Studies on competition in rice. iii. The mechanism of competition among phenotypes. *Evolution* (1968), 529–542.

[77] Jennings, P. R., Coffman, W. R., Kauffman, H. E., et al. Rice improvement. *Rice improvement* (1979).

[78] Jing, F., Ke-feng, Y., Yang, Y., and Ying, T. The simulation of the growing process of virtual plant leaves. In *Audio Language and Image Processing (ICALIP), 2010 International Conference on* (2010), IEEE, pp. 1377–1381.

[79] Jones, C. A., Kiniry, J. R., and Dyke, P. *CERES-Maize: A simulation model of maize growth and development*. Texas AandM University Press, 1986.

[80] Kaufman, A. A. *Statisticheskaya nauka v Rossii: teoriya i metodologiya, 1806–1917 (Statistical science in Russia: theory and methodology, 1806–1917)*. Moscow: TsSU, 1922.

[81] Kikuzawa, K. Phenological and morphological adaptations to the light environment in two woody and two herbaceous plant species. *Functional Ecology 17*, 1 (2003), 29–38.

[82] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., et al. Optimization by simulated annealing. *science 220*, 4598 (1983), 671–680.

[83] Kniemeyer, O. *Design and implementation of a graph grammar based language for functional-structural plant modelling*. PhD thesis, Brandenburg University of Technology Cottbus, 2008.

[84] Kniemeyer, O., Barczik, G., Hemmerling, R., and Kurth, W. Relational growth grammars–a parallel graph transformation approach with applications in biology and architecture. In *Applications of Graph Transformations with Industrial Relevance*. Springer, 2008, pp. 152–167.

[85] Kniemeyer, O., Buck-Sorlin, G., and Kurth, W. GroIMP as a platform for functional-structural modelling of plants. In: Functional-Structural Plant Modelling in Crop Production (eds.: J. Vos, L. F. M. Marcelis, P. H. B. deVisser, P. C. Struik, J. B. Evers). *Proceedings of a workshop held in Wageningen (NL), 5.-8. 3. 2006. Springer, Berlin* (2007), 43–52.

[86] Kniemeyer, O., Hemmerling, R., Streit, K., Ong, Y., and Kurth, W. *Website of GroIMP (Growth-grammar related Interactive Modelling Platform).* http://www.grogra.de, accessed: 25 July 2014, 2014.

[87] Koza, J. R. *Genetic Programming.* MIT Press, 1993.

[88] Koza, J. R. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, 1994.

[89] Kropff, M. J., and Van Laar, H. *Modelling crop-weed interactions.* Int. Rice Res. Inst., 1993.

[90] Kurth, W. Introduction to rule-based programming, l-systems and xl. Summer School 'Modelling and Simulation with GroIMP' / Tutorial for beginners. University of Göttingen (Germany), 27-28 September, 2010.

[91] Kurth, W. Some basic examples in xl (part 1). Summer School 'Modelling and Simulation with GroIMP' / Tutorial for beginners. University of Göttingen (Germany), 27-28 September, 2010.

[92] Kurth, W. Some basic examples in xl (part 2). Summer School 'Modelling and Simulation with GroIMP' / Tutorial for beginners. University of Göttingen (Germany), 27-28 September, 2010.

[93] Kurth, W. *Growth grammar interpreter grogra 2.4-a software tool for the 3-dimensional interpretation of stochastic, sensitive growth grammars in the context of plant modelling. Introduction and Reference Manual*, vol. 32. Berichte des Forschungszentrums Waldökosysteme der Universität Göttingen, 1994.

[94] Kurth, W. Morphological models of plant growth: possibilities and ecological relevance. *Ecological Modelling 75* (1994), 299–308.

[95] Kurth, W. Specification of morphological models with L-systems and relational growth grammars. *Image: Journal of Interdisciplinary Image Science 5* (2007), 25.

[96] Letort, V., Mahe, P., Cournede, P.-H., De Reffye, P., and Courtois, B. Quantitative genetics and functional–structural plant growth models: simulation of quantitative trait loci detection for model parameters and application to potential yield optimization. *Annals of Botany 101*, 8 (2008), 1243–1254.

[97] Li, X., Sudarsanam, N., and Frey, D. D. Regularities in data from factorial experiments. *Complexity 11*, 5 (2006), 32–45.

[98] Litbang-Pertanian. *Varietas unggul*. http://www.litbang.pertanian.go.id/varietas, accessed: 25 August 2015, 2015.

[99] Lo, E., Wang, Z. M., Lechowicz, M., Messier, C., Nikinmaa, E., Perttunen, J., and Sievanen, R. Adaptation of the Lignum model for simulations of growth and light response in jack pine. *Forest Ecology and Management 150*, 3 (2001), 279–291.

[100] Long, S. P., ZHU, X.-G., Naidu, S. L., and Ort, D. R. Can improvement in photosynthesis increase crop yields? *Plant, Cell & Environment 29*, 3 (2006), 315–330.

[101] Lopez, G., Favreau, R. R., Smith, C., Costes, E., Prusinkiewicz, P., and DeJong, T. M. Integrating simulation of architectural development and source–sink behaviour of peach trees by incorporating markov chains and physiological organ function submodels into L-Peach. *Functional Plant Biology 35*, 10 (2008), 761–771.

[102] Luquet, D., Song, Y. H., Elbelt, S., This, D., Clément-Vidal, A., Périn, C., Fabre, D., and Dingkuhn, M. Model-assisted physiological analysis of phyllo, a rice architectural mutant. *Functional Plant Biology 34*, 1 (2007), 11–23.

[103] Ma, Y., Li, B., Zhan, Z., Guo, Y., Luquet, D., de Reffye, P., and Dingkuhn, M. Parameter stability of the structural-functional plant model Greenlab as affected by variation within populations, among seasons and among growth stages. *Annals of Botany 99* (2007), 61–73.

[104] Makarim, A., and Suhartatik, E. Morfologi dan fisiologi tanaman padi. *Balai Besar Penelitian Tanaman Padi Indonesia* (2009).

[105] Makarim, A. K., and Suhartatik, E. *Morphology and Physiology of Rice Plant*. Indonesian center for rice research, Indonesian agency for agricultural research and development (IAARD), Indonesian Ministry of Agriculture, 2011.

[106] Malhotra, R., Singh, N., and Singh, Y. Genetic algorithms: Concepts, design for optimization of process controllers. *Computer and Information Science 4*, 2 (2011), 39–54.

[107] Marcelis, L., Heuvelink, E., and Goudriaan, J. Modelling biomass production and yield of horticultural crops: a review. *Scientia Horticulturae 74*, 1 (1998), 83–111.

[108] Matsubayashi, M., Shikenjō, N. N., and Kōnosu-shi, J. *Theory and practice of growing rice*. Fuji, 1967.

[109] McKay, M. D., Beckman, R. J., and Conover, W. J. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*, 2 (1979), 239–245.

[110] McMillan, J. H. *Educational research: Fundamentals for the consumer*. ERIC, 1996.

[111] Mee, R. *A comprehensive guide to factorial two-level experimentation*. Springer Science & Business Media, 2009.

[112] Mespoulet, M. From typical areas to random sampling: Sampling methods in russia from 1875 to 1930. *Science in Context 15*, 03 (2002), 411–425.

[113] Michalewicz, Z. *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media, 1992.

[114] Michalewicz, Z., and Fogel, D. *How to Solve it (2nd rev. and extended ed.)*. Springer, 2004.

[115] Moldenhauer, K., and Slaton, N. Rice growth and development. *Rice Production Handbook* (2001), 7–14.

[116] Moulia, B., Fournier, M., and Guitard, D. Mechanics and form of the maize leaf: in vivo qualification of flexural behaviour. *Journal of Materials Science 29*, 9 (1994), 2359–2366.

[117] Nakagawa, H., and Horie, T. Modelling and prediction of developmental process in rice. ii. A model for simulating panicle development based on daily photoperiod and temperature. *Japanese Journal of Crop Science 64*, 1 (1995), 33–42.

[118] Nikinmaa, E. Analysis of the growth of Scots pine; matching structure with function (seloste: analyysi männyn kasvusta; rakenteen sopeutumista aineenvaihduntaan). *Acta Forestalia Fennica 235* (1992).

[119] Niklas, K. J. Morphological evolution through complex domains of fitness. *Proceedings of the National Academy of Sciences 91*, 15 (1994), 6772–6779.

[120] Niklas, K. J. Evolutionary walks through a land plant morphospace. *Journal of Experimental Botany 50*, 330 (1999), 39–52.

[121] Niklas, K. J. The evolution of plant body plans—a biomechanical perspective. *Annals of Botany 85*, 4 (2000), 411–438.

[122] Niklas, K. J. Computer models of early land plant evolution. *Annu. Rev. Earth Planet. Sci. 32* (2004), 47–66.

[123] Niklas, K. J., and Kutschera, U. The evolutionary development of plant body plans. *Functional Plant Biology 36*, 8 (2009), 682–695.

[124] Novoplansky, A. Developmental plasticity in plants: implications of non-cognitive behavior. *Evolutionary Ecology 16*, 3 (2002), 177–188.

[125] Perttunen, J., Änen, R. S., Nikinmaa, E., Salminen, H., Saarenmaa, H., et al. Lignum: a tree model based on simple structural units. *Annals of Botany 77*, 1 (1996), 87–98.

[126] Polak, E. *Optimization: Algorithms and Consistent Approximations*, vol. 124. Springer Science & Business Media, 1997.

[127] Pommel, B., Sohbi, Y., and Andrieu, B. Use of virtual 3d maize canopies to assess the effect of plot heterogeneity on radiation interception. *Agricultural and Forest Meteorology 110*, 1 (2001), 55–67.

[128] Prévot, L., Ariès, F., and Monestiez, P. Modélisation de la structure géométrique du maïs. *Agronomie 11*, 6 (1991), 491–503.

[129] Prusinkiewicz, P. Simulation modeling of plants and plant ecosystems. *Communications of the ACM 43*, 7 (2000), 84–93.

[130] Prusinkiewicz, P., Erasmus, Y., Lane, B., Harder, L. D., and Coen, E. Evolution and development of inflorescence architectures. *Science 316*, 5830 (2007), 1452–1456.

[131] Prusinkiewicz, P., Karwowski, R., and Lane, B. *The L+C Plant Modelling Language. Functional-Structural Plant Modelling in Crop Production.* J. Vos, et al., eds. Springer, 2007.

[132] Prusinkiewicz, P., and Lindenmayer, A. *The algorithmic beauty of plants.* Springer Science & Business Media, 1990.

[133] Rao, S. S. *Engineering optimization: theory and practice.* John Wiley & Sons, 2009.

[134] Renton, M. Aristotle and adding an evolutionary perspective to models of plant architecture in changing environments. *Frontiers in Plant Science 4*, 284 (2013), 1–4.

[135] Renton, M., Guédon, Y., Godin, C., and Costes, E. Similarities and gradients in growth unit branching patterns during ontogeny in 'fuji' apple trees: a stochastic approach. *Journal of Experimental Botany 57*, 12 (2006), 3131–3143.

[136] Rey, H., Dauzat, J., Chenu, K., Barczi, J.-F., Dosio, G. A., and Lecoeur, J. Using a 3-d virtual sunflower to simulate light capture at organ, plant and plot levels: contribution of organ interception, impact of heliotropism and analysis of genotypic differences. *Annals of Botany 101*, 8 (2008), 1139–1152.

[137] Rueda, L., and Vidyadharan, V. A hill-climbing approach for automatic gridding of cDNA microarray images. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 3*, 1 (2006), 72–83.

[138] Russell, S., and Norvig, P. *Artificial Intelligence: a Modern Approach (3rd edition)*. Pearson Education, Inc., 1995.

[139] Ryle, G., Cobby, J., and Powell, C. Synthetic and maintenance respiratory losses of 14co2 in uniculm barley and maize. *Annals of Botany 40*, 3 (1976), 571–586.

[140] Sánchez, B., Rasmussen, A., and Porter, J. R. Temperatures and the growth and development of maize and rice: a review. *Global Change Biology 20*, 2 (2014), 408–417.

[141] Schultz, E. A., and Haughn, G. W. Genetic analysis of the floral initiation process (flip) in Arabidopsis. *Development 119*, 3 (1993), 745–765.

[142] Sheehy, J. Limits to yield for C3 and C4 rice: an agronomist's view. *Studies in Plant Science 7* (2000), 39–52.

[143] Sheehy, J. E., and Mitchell, P. Designing rice for the 21st century: the three laws of maximum yield. Tech. rep., IRRI Discussion Paper 48, International Rice Research Institute, Los Banos, 2013.

[144] Shi, Y., and Eberhart, R. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence* (1998), IEEE, pp. 69–73.

[145] Shinozaki, K., Yoda, K., Hozumi, K., and Kira, T. A quantitative analysis of plant form - the pipe model theory: I. Basic analyses. *Japanese Journal of Ecology 14*, 3 (1964), 97–105.

[146] Sitaresmi, T., Yunani, N., Zakki, K., Mulsanti, I. W., Utomo, S. T., and Daradjat, A. Identifikasi varietas contoh untuk karakter penciri spesifik sebagai penunjang harmonisasi pengujian buss padi. *Penelitian Pertanian Tanaman Pangan 32*, 3 (2013), 148–158.

[147] Smoleňová, K., and Hemmerling, R. Growing virtual plants for virtual worlds. In *Proceedings of the 24th Spring Conference on Computer Graphics* (2008), ACM, pp. 67–74.

[148] Smolenová, K., Kurth, W., and Cournède, P.-H. Parallel graph grammars with instantiation rules allow efficient structural factorization of virtual vegetation. *Electronic Communications of the EASST* (2012), 114–128.

[149] Song, Q., Zhang, G., and Zhu, X.-G. Optimal crop canopy architecture to maximise canopy photosynthetic co2 uptake under elevated co2–a theoretical study using a mechanistic model of canopy photosynthesis. *Functional Plant Biology 40*, 2 (2013), 108–124.

[150] Song, Y., Luquet, D., Mathieu, A., De Reffye, P., and Dingkuhn, M. Using Greenlab model to assist to analyse rice morphogenesis: Case of phyllomutant and its wild type 'Nippon bare'. In *Plant Growth Modeling and Applications, 2006. PMA'06* (2006), IEEE, pp. 169–174.

[151] Stein, M. Large sample properties of simulations using latin hypercube sampling. *Technometrics 29*, 2 (1987), 143–151.

[152] Stone, P. The effects of heat stress on cereal yield and quality. *Crop responses and adaptations to temperature stress* (2001), 243–291.

[153] Streit, C. Graphtal user manual. *SIG Computer Graphics, University of Berne, Switzerland* (1992).

[154] Streit, K., and Buck-Sorlin, G. How to develop a simple fspm with groimp. Summer School 'Modelling and Simulation with GroIMP' / Tutorial for beginners. University of Göttingen (Germany), 18 September, 2013.

[155] Suprihatno, B., and Padi, B. B. P. T. *Deskripsi varietas padi*. Balai Besar Penelitian Tanaman Padi, Badan Penelitian dan Pengembangan Pertanian, Departemen Pertanian, 2010.

[156] Tanaka, A. Comparisons of rice growth in different environments. *Climate and Rice. International Rice Research Institute, Los Baños, Philippines* (1976), 429–448.

[157] Thiele, J. C., Kurth, W., and Grimm, V. Facilitating parameter estimation and sensitivity analysis of agent-based models: A cookbook using Netlogo and 'R'. *Journal of Artificial Societies and Social Simulation 17*, 3 (2014).

[158] Troll, W. *Die Infloreszenzen I.* Gustav Fischer Verlag, Jena, Germany, 1964.

[159] Troll, W. *Die Infloreszenzen II.* Gustav Fischer Verlag, Jena, Germany, 1969.

[160] Tsunoda, S. A development analysis of yielding ability in varieties of yield crops. ii. The assimilation-system of plants as affected by the form, direction and arrangement of single leaves. *Japanese Society of Breeding 9* (1959), 237–244.

[161] Tulip, P. R. *Dielectric and lattice dynamical properties of molecular crystals via density functional perturbation theory: implementation within a first principles code.* PhD thesis, Durham University, 2004.

[162] Umeki, K., Kikuzawa, K., and Sterck, F. J. Influence of foliar phenology and shoot inclination on annual photosynthetic gain in individual beech saplings: A functional–structural modeling approach. *Forest Ecology and Management 259*, 11 (2010), 2141–2150.

[163] Utama, D., Ong, Y., Streit, K., and Kurth, W. Determining the influence of plant architecture on light interception of virtual rice plants on the simulation platform groimp. In *Proceedings of the International Conference on Plant Physiology 2014. Enhancing Strategic Plant Physiological Research and Technologies for Sustainable Resources* (2014), MSPP and ICCRI, pp. 92–101.

[164] Valentine, H. *A carbon-balance model of tree growth with a pipe-model framework. In: Dixon RK, Meldahl RS, Ruark GA, Warren WG, eds. Process modeling of forest growth responses to environmental stress.* Portland, Oregon: Timber Press, 1990.

[165] Van Tienderen, P. Morphological variation in Plantago lanceolata: Limits of plasticity. *Evolutionary Trends in Plants 4* (1990), 35–45.

[166] Veach, E. *Robust monte carlo methods for light transport simulation.* PhD thesis, Stanford University, 1997.

[167] Vergara, B. S. Rice plant growth and development. In *Rice.* Springer, 1991, pp. 13–22.

[168] Verhoef, W. Light scattering by leaf layers with application to canopy reflectance modeling: the sail model. *Remote sensing of environment 16*, 2 (1984), 125–141.

[169] Von Mammen, S., and Jacob, C. The evolution of swarm grammars - growing trees, crafting art, and bottom-up design. *Computational Intelligence Magazine, IEEE 4*, 3 (2009), 10–19.

[170] Vos, J., Marcelis, L., and Evers, J. Functional-structural plant modelling in crop production: adding a dimension. *Frontis 22* (2007), 1–12.

[171] Wang, R., Hua, W., Dong, Z., Peng, Q., and Bao, H. Synthesizing trees by pantons. *Visual Computation 22* (2006), 238–248.

[172] Wang, T., and Zhang, L. Image-based fast three-dimensional leaf modeling. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on* (2010), vol. 4, IEEE, pp. V4–525.

[173] Wang, Y., and Li, J. The plant architecture of rice (Oryza sativa). *Plant Molecular Biology 59*, 1 (2005), 75–84.

[174] Warr, P. Food policy and poverty in Indonesia: a general equilibrium analysis. *Australian Journal of Agricultural and Resource Economics 49*, 4 (2005), 429–451.

[175] Watanabe, T., Hanan, J. S., Room, P. M., Hasegawa, T., Nakagawa, H., and Takahashi, W. Rice morphogenesis and plant architecture: measurement, specification and the reconstruction of structural development by 3d architectural modelling. *Annals of Botany 95*, 7 (2005), 1131–1143.

[176] Watson, D. The estimation of leaf area in field crops. *The Journal of Agricultural Science 27*, 03 (1937), 474–483.

[177] Weberling, F. *Morphology of flowers and inflorescences*. CUP Archive, 1992.

[178] Winder, R., and Roberts, G. *Developing Java Software*. John Wiley & Sons, Inc., 1997.

[179] Wood, S. *Generalized additive models: an introduction with R*. CRC press, 2006.

[180] Wopereis, M., Defoer, T., Idinoba, P., Diack, S., and Dugué, M. Curriculum for participatory learning and action research (plar) for integrated rice management (irm) in inland valleys of sub-saharan africa: Technical manual. *Africa Rice Center, Cotonou, Benin* (2009), 26–32.

[181] Xiang, Y., Gubian, S., Suomela, B., and Hoeng, J. Generalized simulated annealing for global optimization: the gensa package. *The R Journal 5*, 1 (2013), 13–27.

[182] YAN, H.-P., KANG, M. Z., DE REFFYE, P., AND DINGKUHN, M. A dynamic, architectural plant model simulating resource-dependent growth. *Annals of Botany 93*, 5 (2004), 591–602.

[183] YOSHIDA, S. *Fundamentals of rice crop science*. Int. Rice Res. Inst., 1981.

[184] ZHAN, Z., WANG, Y., DE REFFYE, P., WANG, B., AND XIONG, Y. Architectural modeling of wheat growth and validation study. In *ASAE Annual International Meeting* (2000).

[185] ZHANG, D., AND LU, G. A comparative study on shape retrieval using Fourier descriptors with different shape signatures. In *Proc. International Conference on Intelligent Multimedia and Distance Education (ICIMADE01)* (2001).

[186] ZHANG, Y., TANG, L., LIU, X., LIU, L., CAO, W., AND ZHU, Y. Modeling morphological dynamics and color characteristics of rice panicle. *European Journal of Agronomy 52* (2014), 279–290.

[187] ZHENG, Z., IWATA, H., HIRATA, Y., AND TAMURA, Y. Quantitative evaluation of the degree of sprout leaf bending of rice cultivars using p-type Fourier descriptors and principal component analysis. *Euphytica 163*, 2 (2008), 259–266.

[188] ZHU, X.-G., LONG, S. P., AND ORT, D. R. Improving photosynthetic efficiency for greater yield. *Annual review of plant biology 61* (2010), 235–261.

[189] ZHU, Y., CHANG, L., TANG, L., JIANG, H., ZHANG, W., AND CAO, W. Modelling leaf shape dynamics in rice. *NJAS-Wageningen Journal of Life Sciences 57*, 1 (2009), 73–81.