Differential Evolution for Parameterized Procedural Woody Plant Models Reconstruction

Aleš Zamuda, Janez Brest, Borko Bošković, Viljem Žumer

Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova ul. 17, 2000 Maribor, Slovenia

Abstract

This paper presents an approach for reconstruction of procedural three-dimensional models of woody plants (trees). The used procedural tree model operates by recursively computing all building parts of a three-dimensional tree structure by applying a fixed procedure on a given large set of numerically coded input parameters. The parameterized procedural model can later be used for computer animation.

Reconstruction of a parameterized procedural model from images is done by differential evolution algorithm which evolves this model by fitting a set of its rendered images to a set of given reference images. The comparison is done on pixel level of the images through the integration of distances to the nearest similar pixels. The obtained results show that the presented approach is viable for modeling of woody plants for computer animation by evolution of the numerically coded procedural model.

Keywords: Woody Plant, Structure Reconstruction, Procedural Model, Differential Evolution, Numerical Encoding

1. Introduction

In this paper[1] we present a new approach to design three-dimensional geometrical models for woody plants (trees). The geometrical models are ex-

Email address: {ales.zamuda,janez.brest,borko.boskovic,zumer}@uni-mb.si (Aleš Zamuda, Janez Brest, Borko Bošković, Viljem Žumer)

URL: http://labraj.uni-mb.si/ (Aleš Zamuda, Janez Brest, Borko Bošković, Viljem Žumer)

[1]Part of this paper was presented at BIOMA 2010 conference [77].

pressed indirectly with the use of procedural models to reduce the enormous data storage space needed for their representation. The procedural models can also be easily animated and are suitable in computer graphics and animation. Our new approach uses evolutionary algorithms to design woody plant models. The proposed approach is based on reconstruction of their procedural models from images. Beaumont and Stepney [7] presented an approach for reconstruction of procedural models. However, their obtained procedural models were not as complex to express woody plants. Also their reconstructed procedural models were two-dimensional. Since, based on our best knowledge, there is no such method for non-interactive three-dimensional tree modeling yet, this is the main contribution of our approach. It is also important to stress that our approach uses fixed dimensionality to encode a tree and is therefore faster than past approaches. Findings and their practical potentials of our work also include, but are not limited to, creation of procedural models from woody plant photographs taken by digital cameras, woody plant images rendered by other software, 3D mesh construction from previously unknown procedural models, animation of growth and windy sway of a sample woody plant, and synthetic movie generation from natural imagery.

In the next section, the related work and background is presented. In the Section 3, the proposed approach for procedural models reconstruction using differential evolution is described. In the Section 4, experimental results and their discussion is given, which show that the given approach is suitable for design of woody plant models. The Section 5 concludes with final remarks and propositions for future work.

2. Related Work and Background

In this section, we present the differential evolution algorithm and one of its best variants, the jDE algorithm [11, 13]. Then, we list some of the procedural models for modeling of trees and outline the numerically coded procedural model of the EcoMod framework [78].

2.1. Procedural Woody Plant Models

For realistic computer visualization of natural environments it is mandatory to include certain level of vegetation (e.g. grass, trees, shrubs). For credible three-dimensional rendering, geometrical descriptions of models are necessary, especially when we want to animate e.g. forest growth [78]. Trees

are the most visible individual elements among vegetation in natural environments. Several techniques for geometry model description and creation exist today. Manual editing of a tree structure and its leaves is a tedious task, since each branch and leaf position, rotation, size, and texture must be appointed. Therefore, procedural tree models are used instead, and several techniques for procedural models are available today. Different procedural models are based on various types of branching structure construction [52]. These techniques differ in the level of detail [4, 8, 55], branch development process [58] flexibility, and pretentiousness of modeling [31, 70], space [49], and time complexity [49] in addition to the animation ability and representation of the built three-dimensional model. The majority of these models try to determine some visible properties of the final three-dimensional model, such as the rotation of branches around their central axes. These properties are usually biologically inspired by *phyllotaxis*, i.e. the main influence on the tree's architecture [56].

Aono and Kunii [4] developed one of the first specialized models for generation of trees. They created four geometry-based models, each being more detailed model of the former. All models have some common rules. The first rule is that every branching of base branch forms two sub-branches. Length and diameter of branches diminish with a constant factor in direction from trunk to smaller branches. Branching angles are equal for all branches in the same level. Plane on which sub-branches lie is perpendicular to the plane between base branch and precedent base branch. Branchings are always carried out on top of existing branches. Their model also describes the angle of axes rotation of successive branches, which is named *phyllotaxis* and dictates most of geometrical structure [56]. Their model uses vertex attractors which attach branches in directions of attractors for wind flow impact, sunrays, and gravity.

Bloomenthal [8] built its model based on user-administered skeleton. His model is only able to add saddles to branches and wrap up the branches with NURBS surfaces and textures. Reeves and Blau [55] used particle systems to represent forest areas with lower detailed models. Branches in their model are represented with straight lines, leaves with vertices or small circles, and base trunks with cones. Parameters of their model are bottom width of crown, height of base trunk, average branch length, and branching angle. Branching structure obtained is subsequently processes with algorithms for simulation of gravity, wind flow, gravity impact, and attraction to light. Oppenheimer [49] modeled his trees using fractal techniques. Base trunk and

branches of their model are built by application of linear transformations defined by 3×3 matrices. Tree geometry consists of prisms or straight lines with textures. These models were rendered for high-detail usage, but did not include any leaves. Strnad and Guid [62] represented trees at middle level of detail with fractal hyper textures. Their tree model is based on fractal iterative functional systems which are visualized using volume rendering. Because of its time-consuming rendering technique, this model was not applicable for real-time animation.

Prusinkiewicz and Lindenmayer [52] created trees using biologically inspired models with L-systems [35] and they were applicable for high detail usage. They introduced visual representation of rewriting strings into L-systems, which are composed using context sensitive grammars. Visualization of this strings is similar to LOGO turtle [50] principle, which draws geometrical elements guided by the string program, representing topology of a tree. This technique was very flexible and several improvements were made on it. The disadvantage of this technique is pretentious realization of a good modeler and therefore also the modeling itself, which requires users to know the domain specific language to define grammars [42] for rewrite rules. Interactive modelers are emphasized in following years for the technique of Prusinkiewicz and Lindenmayer by Deussen02 et al. [23, 36].

Holton [31] created trees with the use of biologically inspired strand model. An advantage of this model is that, thickness of branches and proportions between branching angles are determined directly with internal rules in the model. Strands flow along branches and are divided without splitting a single strand. Branches with single strands are carrying leaves. Strand distribution determines thicknesses and lengths of branches. User enters the number of strands along the tree, proportions between branch lengths, and branching angles to parameterize the procedural model. Certain attractors influence the branching structure, e.g. central trunk uprightness, gravimorphism, phototropism, planartropism, and phyllotaxis. A downside of the model is that user still has to enter a huge amount of numerical data, which diminishes the flexibility of the model.

Weber and Penn [70] represented the tree model with the use of simple geometry without a development of branching topology. For all branches at the same level, they entered a branching angle, branch length proportions and thickness for branches. Their model introduces wind sway animation, branch cutting to predetermined volume, and progressive level of detail rendering.

All listed techniques require a compromise between flexibility and ease

of use. Some general-purpose 3D modelers include specialized tools for construction of trees. An unsolved issue of these tools is that they are usually not flexible enough to model any tree model and usually allow modeling of only predefined families of tree species. If they are flexible, they are usually too flexible, do not allow user to enter data graphically but only numerically, and the parameters are poorly described. In response to this issue a well understood, simple, and user friendly EcoMod tool to design tree models has been designed [79]. The EcoMod framework incorporates a procedural model for woody plants, based on the Holton and Webber-Penn models. The procedural model and its modeler with woody plant models library is described by Zamuda et al. [78, 79]. The tool enables parallel construction of trees on a terrain, where trees can be put side by side, and the user can shape tree models simultaneously. The trees designed with this model can be foliage or coniferous trees with very different branching structures. Each branch and each leaf can be animated in real time to show the growth of a tree or its sway in the wind. By slightly modifying the parameters of procedural models, it can achieve computer animation of these models [79], thereby creating a large number of geometrical models from a single procedural model. The modeler for natural trees is integrated in a framework for animation and simulation of forest ecosystems growth. The EcoMod tool is a publicly available[2] free open source software. The software is also parallelized to exploit all available computer resources, such as multi-core, grid, and GPU [82].

2.2. Image-based Approaches to Modeling

Image-based approaches have the best potential to produce realistically looking plants, since they rely on images of real plants [60, 48, 19, 54]. Also, little work has been done to design trees from images without user interaction using EA-based reconstruction. Beaumont and Stepney [7] presented an approach for reconstruction of procedural models. However, the procedural models used were two-dimensional. Therefore, we extended their approach to the domain of three-dimensional procedural models suitable to model woody plants without user interaction.

2.3. Differential Evolution

Differential Evolution (DE) [61] is a floating-point encoding evolutionary algorithm [30, 25, 82] for global optimization over continuous spaces, which

[2]http://sourceforge.net/projects/ecomod/, http://ecomod.sourceforge.net

can also work with discrete variables. Its main performance advantages over other evolutionary algorithms [11, 43, 71, 38] lie in floating-point encoding and a good combination of evolutionary operators, the mutation step size adaptation, and elitist selection. The DE algorithm has a main evolution loop in which a population of vectors is computed for each generation of the evolution loop. During one generation G, for each vector $\mathbf{x}_i$, $\forall i \in \{0, NP\}$ in the current population, DE employs evolutionary operators, namely mutation, crossover, and selection, to produce a trial vector (offspring) and to select one of the vectors with the best fitness value. NP denotes population size and G the current generation number.

Mutation creates a mutant vector $\mathbf{v}_{i,G+1}$ for each corresponding population vector. Among many proposed, one of the most popular DE mutation strategies is the *'rand/1/bin'* [51, 61]:

$$\mathbf{v}_{i,G+1} = \mathbf{x}_{r_1,G} + F(\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}),$$

where the indexes r_1, r_2, and r_3 represent the random and mutually different integers generated within the range $\{1, NP\}$ and also different from index i. F is an amplification factor of the difference vector within the range $[0, 2]$, but usually less than 1. Vector at index r_1 is a base vector. The term $\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}$ denotes a difference vector which after multiplication with F, is named amplified difference vector.

After mutation the mutant vector $\mathbf{v}_{i,G+1}$ is taken into recombination process with the target vector $\mathbf{x}_{i,G}$ to create a trial vector $u_{i,j,G+1}$. The binary crossover operates as follows:

$$u_{i,j,G+1} = \begin{cases} v_{i,j,G+1} & \text{if } rand(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{i,j,G} & \text{otherwise} \end{cases},$$

where $j \in \{1, D\}$ denotes the j-th search parameter of D-dimensional search space, $rand(0,1) \in [0,1]$ denotes a uniformly distributed random number, and j_{rand} denotes a uniform randomly chosen index of the search parameter, which is always exchanged to prevent cloning of target vectors. CR denotes the crossover rate [73].

Finally, the selection operator propagates the fittest individual [20] in the new generation (for minimization problem):

$$\mathbf{x}_{i,G+1} = \begin{cases} \mathbf{u}_{i,G+1} & \text{if } f(\mathbf{u}_{i,G+1}) < f(\mathbf{x}_{i,G}) \\ \mathbf{x}_{i,G} & \text{otherwise} \end{cases}.$$

DE was proposed by Storn and Price [61] and since then, it has been modified and extended several times with new versions proposed [51, 27, 37, 26, 3, 75, 53, 80, 67, 44, 21, 45, 39, 47, 22] and its derivations have won several evolutionary algorithm competitions [32, 81, 13]. DE was also introduced for multiobjective optimization [1, 57, 74, 76, 66]. Several evolutionary algorithms have been proposed, to list just a few, see [28, 6, 29, 38, 83, 34, 82]. Performance of various DE variants has widely been studied and compared to other evolutionary algorithms, also on various competitions at major scientific conferences [32, 63, 22]. In the survey by Neri and Tirronen [47], it has been concluded compared to the other algorithms, that jDE is superior in terms of robustness and versatility over diverse benchmark set. The jDE [11] has also won 2009 WCCI Dynamic and Uncertain Environments competition [13] and performed well on other competitions on Large Scale Global Optimization in 2008 and 2010 [14, 16]. Therefore, we choose to apply jDE in this paper for an optimizer.

Successful DE applications have also been published in several major journals. Joshi and Sanderson [33] have used DE for minimal representation multisensor fusion. Chang and Chang [18] used DE to reduce harmonic voltage distortion in electrical distribution systems. Varadarajan et al. [68] considered DE for reactive power dispatch. Maulik et al. [41] applied DE for pixel classification. Neri and Mininno [46] have applied DE to robot control by introducing memetic operators and compact representation. Salvatore et al. [59] used DE to optimize an algorithm for sensorless induction motor control. Tušar et al. used their DEMO optimizer to parameterize an electric motor design [66]. Tirronen et al. applied DE to paper production defects detection [65]. Alatas et al. have mined numeric association rules in optimization of multi-objective problems [2]. Bošković et al. used history mechanism modified DE to parameterize computer chess engine evaluation function metrices [5]. Caponio et al. [17] analyzed super-fit control adaptation in memetic DEs.

The original DE algorithm keeps all three control parameters fixed during the optimization process. However, there still exists a lack of knowledge how to find reasonably good values for the control parameters of DE, for a given function [37, 64, 24]. Based on the experiment in [11], the necessity of changing control parameters during the optimization process was confirmed. The Self-Adaptive DE (jDE) from [11] refers to the self-adapting mechanism on the control parameters. The self-adapting mechanism uses the already exposed '*rand/1/bin*' strategy. The self-adaptive control mechanism is used

to change the control parameters F and CR during the evolutionary process. The third control parameter NP is kept unchanged in [11].

Each individual in the jDE population is extended using the values of these two control parameters. Both of them are applied at individual level. The better values for these (encoded) control parameters lead to better individuals which, in turn, are more likely to survive and produce offspring and, hence, propagate these better parameter values. New control parameters $F_{i,G+1}$ and $CR_{i,G+1}$ are calculated as follows [11]:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \times F_u & \text{if } rand_2 < \tau_1, \\ F_{i,G} & \text{otherwise,} \end{cases}$$

$$CR_{i,G+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2, \\ CR_{i,G} & \text{otherwise.} \end{cases}$$

They produce control parameters F and CR in a new vector. The $rand_j \in [0, 1]$, $j \in \{1, 2, 3, 4\}$ are uniform random values. τ_1 and τ_2 represent the probabilities of adjusting control parameters F and CR, respectively. τ_1, τ_2, F_l, F_u are taken fixed values as proposed in [11]. The new F takes a value from $[0.1, 1.0]$ in a random manner. The new CR takes a value from $[0, 1]$. $F_{i,G+1}$ and $CR_{i,G+1}$ are obtained before the mutation is performed. So they influence the mutation, crossover, and selection operations of the new vector $\mathbf{x}_{i,G+1}$. For this paper, only the original jDE algorithm [11] is used, although the algorithm also has some extensions that are not utilized and we refer the reader to [10, 15, 12, 9, 13, 16].

3. Differential Evolution for Reconstruction of Procedural Woody Plant Models

We combine the jDE algorithm [11] and the numerically coded procedural model of woody plants from EcoMod framework [78]. Thereby, we reconstruct three-dimensional woody plant models from images by evolving the parameters of the procedural model. Fitness computation of an evolved model is based on comparison of two-dimensionally rendered images. The fitness is better (i.e. takes smaller values) for images with greater similarity. The reconstruction method operates by encoding the parameters of the procedural model in genotype of the individual vector of jDE population. In the following, proposed parts of the optimization procedure are described, i.e. the genotype encoding, genotype-phenotype mapping, and its fitness evaluation.

3.1. Genotype Encoding

The procedural woody plant model helps designing a tree by minimizing the set of parameters that a user needs to set. Once these parameters are given, the model automatically determines positions, rotations, sizes and textures for several thousand branch segments and several thousand leaves. An individual tree species model is created by parameterizing the procedural model. It generates a three-dimensional structure [72] of a tree by recursively executing a fixed procedure over a given set of numerically coded input parameters, such as branch thickness, relative branch length and branching structure proportions. These procedural model input parameters are called a breeder [69, 40], since the full morphology of a woody plant is recursively developed from this gene. Each step of the procedure adds a building block of a tree to the geometrical model.

Parameters of EcoMod woody plant procedural model are distinguished as vectors (local) and scalars (global). Global parameters are constant for all branch segments although local parameters vary along Gravelius (g) and Weibull (w) branch order. Branch order (g, w) dependent local parameters are determined for all orders (g, w), where $g \in [0, 15]$, $w \in [0, 50]$ are integer numbers and each 750 Gravelius and Weibull ordered real-coded parameters encode one vector (matrix) parameter. Vector parameters design the strand distribution, branching angles, branch segment proportions, and gravity impact to tree geometry. Scalar parameters of the model are height and thickness of base trunk, wind impact, and density and size of leaves. Using the listed vector and scalar parameters, a geometrical model is built recursively. A visual tool for interactively addressing the many vector parameters was presented in [78] using line-segment addressable parameter convolutions. The EcoMod procedural model differs from many other models [4, 8, 31, 55, 48] in that all of its parameters are fully numerically encoded and are fixed dimensionality. It is therefore especially suitable for parameter estimation using differential evolution.

An individual genotype vector $\mathbf{x}_i$, $i \in \{1..NP\}$ of jDE population, represents the set of procedural model parameters of a breeder. The dimensionality of evolved floating-point encoded breeder is $D = 4509$, compounding $\mathbf{x}_i = \{x_{i,j}\} \in [0, 1]$. The index j denotes a component index in the genotype and it runs for the vector parameters as $j = o + (50 * g + w)$, with o denoting an offset index where a vector parameter encoding begins in the genotype. Components of $\mathbf{x}_i$ encode the following parameters:

- number of strands in a tree $S = 400x_{i,0} + 10 \in [10, 410]$ (determines complexity of a tree, and is also equal to number of leaves creations, branch segments less one, and half branches less two),
- height of base trunk $l_0^{0,0} = 10x_{i,1} \in [0 \text{ m}, 10 \text{ m}]$,
- coefficient of branch thickness $k_d = 0.05x_{i,2} \in [0,\ 0.05]$,
- phyllotaxis angle $\alpha_p = 360x_{i,3} \in [0°, 360°]$,
- branching ratio of subbranch strands distribution $k_s^{g,w} = 0.5x_{i,j} + 0.5$, $\forall j \in [4, 753]$, $k_s^{g,w} \in [\frac{1}{2},\ 1]$,
- branching angle between dividing sub-branches $\alpha^{g,w} = 180x_{i,j}$, $\forall j \in [754, 1503]$, $\alpha^{g,w} \in 0°, 180°$,
- maximum relative sub-branch to base branch length $M^{g,w} = 20x_{i,j}$, $\forall j \in [1504, 2253]$, $M^{g,w} \in [0, 20]$,
- minimum relative sub-branch to base branch length $m^{g,w} = 20x_{i,j}$, $\forall j \in [2254, 3003]$, $m^{g,w} \in [0, 20]$,
- branch length scaling factor $k_l^{g,w} = 20x_{i,j}$, $\forall j \in [3004, 3753]$, $k_l^{g,w} \in [0, 20]$,
- gravicentralism impact $k_c = x_{i,3754} \in [0, 1]$,
- gravimorphism impact (i.e. gravitational bending of branches) $\alpha_m^{g,w} = 360x_{i,j} - 180$, $\forall j \in [3755, 4504]$, $\alpha_m^{g,w} \in [-180°, 180°]$,
- enabling leaves display on a tree $B_l = x_{i,4505} < 0.5?0 : 1 \in \{0, 1\}$,
- size of leaves $l_l = 0.3x_{i,4506} \in [0,\ 0.3]$,
- density of leaves $\rho_l = 30x_{i,4507} \in [0, 30]$, and
- leaf distribution type $l_{type} = 5x_{i,4508}$ of values *Spiral*, *Stacked*, *Staggered*, *Bunched*, or *Coniferous*.

Using this encoding, we are able to obtain several different procedural models for trees [78]. To obtain the beech tree in Figure 1 for the fourth tree ($S = 400$) the breeder parameters are as follows. The scalar procedural model parameters are: $S = 400$, $L_0 = 5$, $k_d = 0.01$, $\alpha_p = 85$,

Figure 1: Rendered procedural models of a beech tree with different number of strands, wireframe tree skeleton without leaves, and, rightmost, the same tree model with different branching structure.

$k_c = 0.3$, $l_l = 0.1$, $\rho_l = 5$, $l_{type} = \textit{Spiral}$, $w_g = 1.5$, $\mathbf{w} = [0\ 1\ 0]^T$, $k_f = 1$ and $l_{LOD} = 0$. To present the matrices easier, we will present them using auxiliary parameters which express the actual parameters using control points. Interactively defining control points of poly-line for a graph specifies values of auxiliary parameters. The values at ordinate of the graph are arbitrary floating point numbers and are sampled for g and w indices at abscissa. For the branch distribution $k_s^{g,w} = \max\left\{\min\left\{k_s^g k_s^w, 1\right\}, \frac{1}{2}\right\}$, the values $k_s^g \in [\frac{1}{2}, \quad 1]$ and $k_s^w \in [0, 2]$ are $\mathbf{k}_s^g = \{(0, 0.75),\ (15, 0.9)\}$ and $\mathbf{k}_s^w = \{(0, 1),\ (50, 1)\}$. For the branching angle between dividing subbranches, $\alpha^{g,w} = \min\left\{\alpha^g \alpha^w, 180°\right\}$, the values $\alpha^g \in [0°, 180°]$ and $\alpha^w \in [0, 2]$ are $\boldsymbol{\alpha}^g = \{(0, 45°),\ (15, 45°)\}$ and $\boldsymbol{\alpha}^w = \{(0, 1),\ (50, 1)\}$. For the maximum relative branch length, $M^{g,w} = M^g M^w$, the values $M^g \in [0, 10]$ and $M^w \in [0, 2]$ are $\mathbf{M}^g = \{(0, 1),\ (15, 1)\}$ and $\mathbf{M}^w = \{(0, 1),\ (50, 1)\}$. For the minimum relative branch length, $m^{g,w} = m^g m^w$, the values $m^g \in [0, 10]$ and $m^w \in [0, 2]$ are $\mathbf{m}^g = \{(0, 1),\ (7.5, 0.45),\ (15, 1)\}$ and $\mathbf{m}^w = \{(0, 1),\ (50, 1)\}$. For the branch length scaling factor, $k_l^{g,w} = k_l^g k_l^w$, the values $k_l^g \in [0, 10]$ and $k_l^w \in [0, 2]$ are $\mathbf{k}_l^g = \{(0, 1),\ (15, 1)\}$ and $\mathbf{k}_l^w = \{(0, 1),\ (50, 1)\}$. For the gravimorphism impact, $\alpha_m^{g,w} = \max\left\{\min\left\{\alpha_m^g \alpha_m^w, 180°\right\}, -180°\right\}$, the values $\alpha_m^g \in [-180°, 180°]$ and $\alpha_m^w \in [0, 2]$ are $\boldsymbol{\alpha}_m^g = \{(0, 0°),\ (15, 0°)\}$ and $\boldsymbol{\alpha}_m^w = \{(0, 1),\ (50, 1)\}$. For leaves, parameters are $B_l = 1$, $\rho_l = 5$, $l_l = 0.1$, $l_{type} = \textit{Spiral}$. The first four trees from left in Figure 1 have only the parameter $S = \{10, 50, 100, 400\}$ different. The fifth tree has all parameters same as the fourth, but it is rendered wireframe and without leaves ($B_l = 0$). The rightmost tree has $k_s^g = \{\ (0, \quad 0.5),\ (15, \quad 0.9)\}$ different, compared to the fourth tree.

3.2. Genotype-phenotype Mapping

Our reconstruction method is based on reconstruction of two-dimensional images of woody plants $\mathbf{z}^* = \{z^*_{x,y}\}$, $\forall x = 0..X–1$, $\forall y = 0..Y–1$, possibly taken by a digital camera. To compare a three-dimensional tree evolved with the use of a genotype $\mathbf{x}$ to the reference image $\mathbf{z}^*$, the encoded D-dimensional genotype $\mathbf{x}$ must be transformed to its phenotype first. Phenotype is a set of rendered two-dimensional images $\mathbf{z}_i = \{z^i_{x,y}\}$, $\forall x = 0..X-1$, $\forall y = 0..Y-1$. A phenotype is computed by rendering its genotype $\mathbf{x}$ which presents a breeder input to the parameterized procedural model for a tree in EcoMod. A geometrical model and a phenotype image is calculated using Algorithm 1. For building particles of a tree, the rendering procedure uses geometrical interpolations for vertex computations and photo textures for pixel colors.

Images $\mathbf{z}^*$ and $\mathbf{z}_i$ are all of dimensionality $X \times Y$ pixels, where the reference image is scaled to the given resolution, if necessary. Both images are converted to black and white, where white (0) pixels mark background and black (1) pixels mark the material, e.g. wood. With the use of the conversion, the evolved procedural model is compared twice to the reference images, differing by camera view angle of $\beta = 90°$ along the trunk base. The latter is done to favor three-dimensional procedural models generation. If we denote the Algorithm 1 as function $\mathbf{g}$ then $\mathbf{z}_i = \mathbf{g}(\mathbf{x}, \beta_i)$, $i = \{1, 2\}$ where $\beta_1 = 0°$ and $\beta_2 = 90°$.

3.3. Phenotype and Reference Image Comparison

Phenotype evaluation is expressed by reconstruction error. We measure the reconstruction error by sum of differences in the reference original image and the generated rendered images of evolved parameterized procedural models:

$$f(\mathbf{x}) = f(\mathbf{g}(\mathbf{x}, \beta_1), \mathbf{g}(\mathbf{x}, \beta_2)) = h(\mathbf{z}_1) + h(\mathbf{z}_2).$$

To calculate differences of these images we chose to compare the images pixel-wise as follows. For each pixel rendered as non-background in the evolved image, we compute the Manhattan distance to the nearest non-background pixel in the reference image, and vice-versa [7]. The sum of these distances adds to the fitness evaluation:

$$h(\mathbf{z}_i) = \sum_{x,y} m_1(z^i_{x,y}, z^*_{x,y}) + \sum_{x,y} m_1(z^*_{x,y}, z^i_{x,y}),\ i \in \{1, 2\},$$

where m_1 denotes a function which computes a Manhattan distance to the nearest pixel in an image $\mathbf{z}^*$, being set to 1 (i.e. black, wood material).

Algorithm 1 Procedural tree model geometrical structure calculation. Recursive procedure is called using `branchsegment`(0, 0, S, 1, $l_0^{0,0}$, $\mathbf{I}$, $\mathbf{I}$), $\mathbf{I}$ denoting an identity matrix.

procedure `branchsegment`(g, w, S_0, L_0, l_0, $\mathbf{M}_0$, $\mathbf{M}_{m;0}^{-1}$)

Require: g, w - Gravelius and Weibull index of base branch; S_0 - number of strands in base branch; L_0, l_0 - base branch relative and actual length; $\mathbf{M}_0$ - base branch coordinate system; $\mathbf{M}_{m;0}^{-1}$ - inverse matrix of rotations for gravimorphism in coordinate system for base branch; global (i.e. part of *breeder*) k_d, k_c, l_{type}, $k_s^{g,w}$, $M^{g,w}$, $m^{g,w}$, $k_l^{g,w}$, $\alpha_m^{g,w}$, $\alpha^{g,w}$, t, k_f, w_s, w_g

Ensure: rendered tree image

$d := k_d\sqrt{S_0}$; {thickness calculation from Mandelbrot}

`render base branch`($\mathbf{M}_0$, l_0, d);

if $S_0 = 1$ **then**

`render leaves`(l_{type}); **return**;

end if

$S_1 := \lceil 1 + k_s^{g,w}(S_0 - 2) \rceil$, $S_2 = S_0 - S_1$; {number of strands in subbranches}

$r_1 := \max\left\{\min\left\{\sqrt{\frac{S_1}{S_0}}, M^{g,w}\right\}, m^{g,w}\right\}$; {branch length proportions based on strands}

$r_2 := \max\left\{\min\left\{\sqrt{\frac{S_2}{S_0}}, M^{g,w}\right\}, m^{g,w}\right\}$;

$L_1 := r_1 L_0$, $L_2 := r_2 L_0$; {relative lengths of subbranches}

$l_1 := k_l^{g,w} L_1$, $l_2 := k_l^{g,w} L_2$; {active subbranch lengths}

$\alpha_1 := k_c\sqrt{\frac{S_2}{S_0}}\alpha^{g,w}$, $\alpha_2 := \alpha^{g,w} - \alpha_1$; {branching angles}

$\mathbf{M}_1 := \mathbf{R}_z(\alpha_1)\mathbf{R}_y(\alpha_p)\mathbf{R}_{\mathbf{y}\times\mathbf{y}_m}(\alpha_m^{g,w})\mathbf{T}_y(l_0)\mathbf{M}_0$; {Translation and rotation matrices}

$\mathbf{M}_2 := \mathbf{R}_z(\alpha_2)\mathbf{R}_y(\alpha_p)\mathbf{R}_{\mathbf{y}\times\mathbf{y}_m}(\alpha_m^{g,w})\mathbf{T}_y(l_0)\mathbf{M}_0$;

$\mathbf{M}_{m;1}^{-1} := \mathbf{R}_{\mathbf{y}\times\mathbf{y}_m}(-\alpha_m^{g,w})\mathbf{R}_y(-\alpha_p)\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_1 - \alpha_z(t))\mathbf{M}_{m;0}^{-1}$; {refreshing inverse matrix for construction of gravimorphism vector}

$\mathbf{M}_{m;2}^{-1} := \mathbf{R}_{\mathbf{y}\times\mathbf{y}_m}(-\alpha_m^{g,w})\mathbf{R}_y(-\alpha_p)\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_2 - \alpha_z(t))\mathbf{M}_{m;0}^{-1}$;

`branchsegment`($g+1$, $w+1$, S_2, L_2, l_2, $\mathbf{M}_2$, $\mathbf{M}_{m;2}^{-1}$); {minor branch development}

`branchsegment`(g, $w+1$, S_1, L_1, l_1, $\mathbf{M}_1$, $\mathbf{M}_{m;1}^{-1}$); {major branch development}

return; {from recursive procedure call}

The whole algorithm outline of *differential evolution for reconstruction of procedural woody plant models* is given in Algorithm 2. Here, phenotype evaluation by Algorithm 1 is wrapped by two DE loops. After termination of these DE loops, the algorithm returns a single parameterized procedural model as a solution.

4. Experimental Results

We have assessed the algorithm for tree reconstruction on an example tree, seen in Figure 3 on the far right. The maximal number of fitness evaluations (FEs) for jDE algorithm was 10000 which equals to [7]. The reference image

Algorithm 2 Reconstruction of a parameterized procedural 3D model of a woody plant from an image.

procedure reconstruction($\mathbf{z}^*$)

Require: S_0 - maximum number of strands in base branch; also, other default parameters for jDE [11] and EcoMod [78]

Ensure: reconstructed parameterized procedural 3D model of a woody plant

`uniform randomly generate DE initial population` $\mathbf{x}_{i,0} \in [0,1]$ `for` $i = 1..NP$;

for DE generation loop g (while FEs < 10000) **do**

for DE iteration loop i (for all individuals $\mathbf{x}_{i,g}$ of a population) **do**

DE individual $\mathbf{x}_{i,g}$ creation (adaptation, mutation, crossover):

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \times F_u & \text{if } rand_2 < \tau_1, \\ F_{i,G} & \text{otherwise} \end{cases};$$

$$CR_{i,G+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2, \\ CR_{i,G} & \text{otherwise} \end{cases};$$

$$\mathbf{v}_{i,G+1} = \mathbf{x}_{r_1,G} + F_{i,G+1}(\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G});$$

$$u_{i,j,G+1} = \begin{cases} v_{i,j,G+1} & \text{if } rand(0,1) \leq CR_{i,G+1} \text{ or } j = j_{rand} \\ x_{i,j,G} & \text{otherwise} \end{cases};$$

DE fitness evaluation (genotype-phenotype mapping, rendering, and comparison):

$\mathbf{z}_1 = \mathbf{g}(\mathbf{u}_{i,g}, \beta_1)$, $\mathbf{z}_2 = \mathbf{g}(\mathbf{u}_{i,g}, \beta_2)$ {Execute Algorithm 1 twice per genotype}

$h(\mathbf{z}_1) = \sum_{x,y} m_1(z^1_{x,y}, z^*_{x,y}) + \sum_{x,y} m_1(z^*_{x,y}, z^1_{x,y})$; {Difference metric}

$h(\mathbf{z}_2) = \sum_{x,y} m_1(z^2_{x,y}, z^*_{x,y}) + \sum_{x,y} m_1(z^*_{x,y}, z^2_{x,y})$; {90° rotation for 3D}

$f(\mathbf{u}_{i,g}) = f(\mathbf{g}(\mathbf{u}_{i,g}, \beta_1), \mathbf{g}(\mathbf{u}_{i,g}, \beta_2)) = h(\mathbf{z}_1) + h(\mathbf{z}_2)$; {Fitness evaluation}

DE selection:

$$\mathbf{x}_{i,G+1} = \begin{cases} \mathbf{u}_{i,G+1} & \text{if } f(\mathbf{u}_{i,G+1}) < f(\mathbf{x}_{i,G}) \\ \mathbf{x}_{i,G} & \text{otherwise} \end{cases};$$

end for

end for

return the best individual obtained;

was taken from a sample evolutionary run from preliminary testing. The sampling rate dimension of the rendered parameterized procedural model was set to 250x250 and the maximum number of strands in the tree to $S = 410$, which far exceeds representable branching complexity of [7] where only four iterations over a quite simple overwrite rule and a fixed branching angle were used to represent structures. The remaining parameters were kept as defaults in original algorithms from their literature.

Final solutions obtained over 30 independent runs for different settings of NP in the evolutionary algorithm jDE are seen in Table 1 and Figure 2. The best average final solution is obtained using $NP = 100$ with fitness of 3031.9 and standard deviation of 184.7. Worst final fitnes for any of the jDE

Table 1: Obtained results for DE and jDE on various population sizes.

NP	Best		Worst		Average		Std. dev.	
	DE	jDE	DE	jDE	DE	jDE	DE	jDE
50	4822	2898	39950	3664	16328.7	3044.1	49626.8	982.2
100	5243	2964	34872	3098	13525.2	**3031.9**	39006.6	184.7
150	6160	3073	31981	3300	14092.8	3175.7	38177.4	309.7
200	6049	3113	20434	3585	10637.4	3310.4	19630.9	608.8
250	6160	3243	17301	3634	10424.9	3400.0	17039.1	540.1
300	5676	3272	17343	4022	9666.6	3580.6	15389.1	915.9
350	4850	3341	21050	4035	9733.6	3660.2	20290.5	1017.3
400	5554	3405	21050	4207	9617.8	3829.0	18981.2	1047.0
450	5554	3450	17796	4364	9111.2	3874.1	13917.7	1281.7
500	5388	3380	16051	4379	**8619.4**	4001.6	12006	1475.3

runs is 4379 for $NP = 500$. The sampled procedural models using $NP = 100$ for run 1, with seed 1, for jDE are seen in Figure 3. The tree on the image is 2.5 m tall, 1 m for the first branch segment; therefore it only extends to a part of the image's canvas which is 25 m tall in total. Therefore, the images, which have original resolution of 250x250, were zoomed in Figure 3 to show only the tree part of the image. As it can be seen in Figure 3, the chosen resolution is high enough for procedural tree model reconstruction.

The best tree obtained at the end has $S_0 = 43$ strands, equal to the tree on the original image. The best image's gene encodes $\alpha_p = 260°$ for phyllotaxis angle and for the original used it was 262°. As an example, these two parameters match in the genotypes quite well. Also, as seen in Figure 4, the images display visually quite similar trees, regarding the metric used. This shows that the chosen maximum strand number $S = 410$, the sampling rate dimension of 250x250, and other parameters setting used can obtain a promising optimization result using our approach.

The evolutionary run convergence for jDE of the best, worst, and median runs for all *NP* is seen in Figure 5. For the best run, the algorithm converges quite fast from fitness of 107553 down to 2964, gaining most optimizations even before 4000 FEs, which implicates 10000 FEs is enough for our optimization. Also, as it is clearly seen in Figure 6 for jDE at $NP = 100$ in additional experiment with maximum 100000 FEs, no much improvement is gained after the original FEs limitation, regarding the metric used.

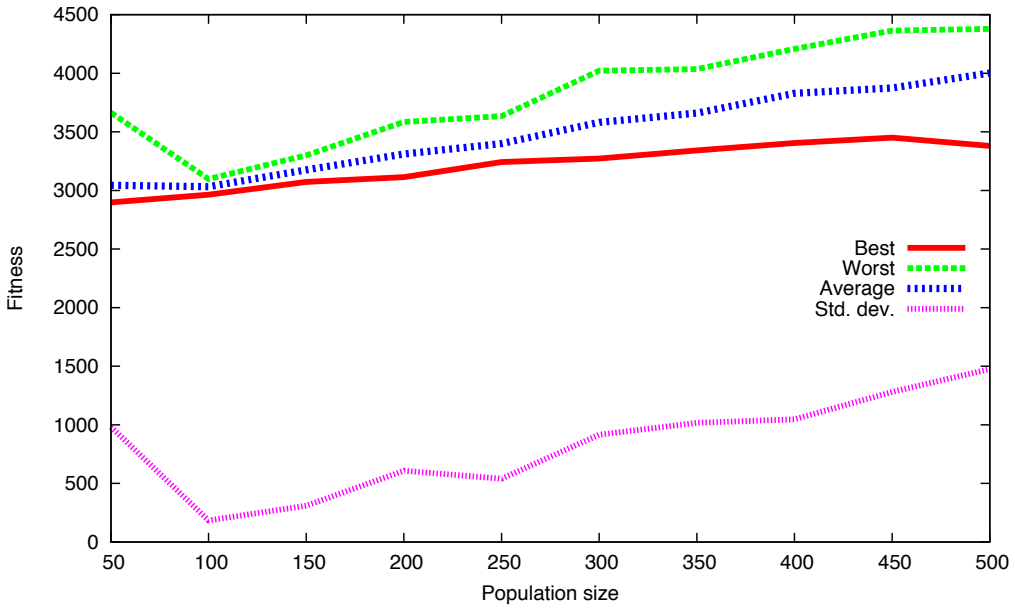

Figure 2: Algorithm jDE performance by obtained fitness, dependent on the population size.

For purpose of statistical performance comparison of our approach with jDE, we have used the original DE with fixed control parameters, $F = 0.5$, $CR = 0.9$, and strategy DE/rand/1/bin. Performance of this optimizer is also seen in Table 1 and plotted in Figure 7. As seen, the performance, as assessed by the metric used, is far worse for DE than is for our jDE. Best obtained fitness with DE is 4822 at $NP = 50$. The best average fitness with DE is 8619.43, obtained for $NP = 500$, where the standard deviation is 12006. As can be read from Table 1 and comparing Figures 2 and 7, the

Figure 3: Rendered evolved parameterized procedural models at FEs $= \{1, 8, 18, 1992, 2727, 3230\}$ and rightmost, the reference image.

Figure 4: The best obtained individual (left) and the reference image (right).

best results of DE are always worse than the worst results of jDE for all population sizes. Therefore, significant performance differences among DE and jDE further suggests benefitial use of jDE on this optimization problem.

Since we can obtain such models as seen in Figure 3 and design woody plants with performance reliability, presented in Table 1, we can conclude that the presented approach is viable for modeling of woody plants for computer animation by evolution of the numerically coded procedural model.

5. Conclusions

We presented an approach to design woody plant geometrical models. To construct a geometrical model, we have used a parameterized procedural model. The parameters of the procedural model were evolved through the jDE differential evolution algorithm. The sampled procedural models were rendered with the use of the EcoMod framework that encodes the tree parameters by fixed dimensionality. Rendered images were then compared to the reference source images, for reconstruction, to guide the optimization process. After the description of the proposed approach, we demonstrated its experimental results by reconstruction of a sample woody plant model

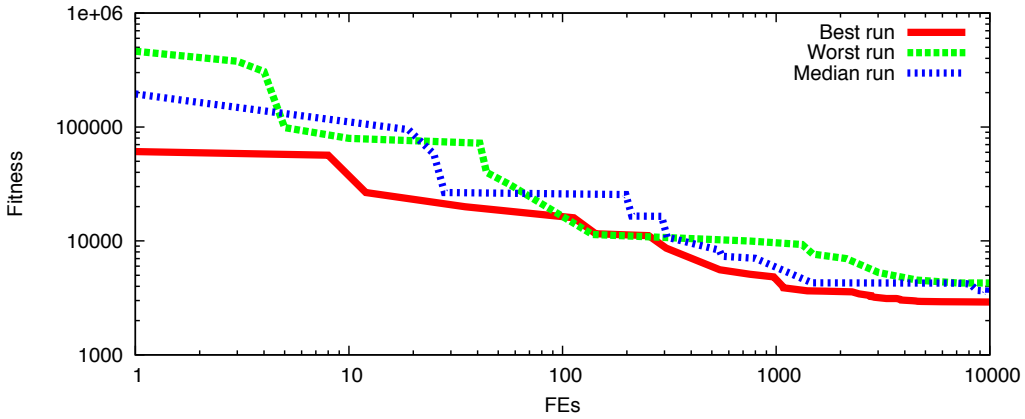

Figure 5: Best, median, and worst evolutionary runs convergence for jDE.

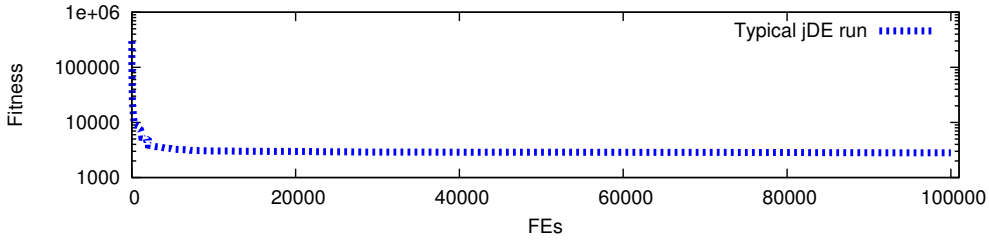

Figure 6: Convergence of a typical jDE run with $NP = 100$, extended to 100000 FEs.

and statistical analysis of the obtained results.

The main advantages of our approach over other approaches [7, 60, 48, 19] are, that it uses fixed dimensionality encoding for trees and thus allows differential evolution, allows three-dimensional parameterized models reconstruction, the models obtained can express greater branching complexity than the existing approach using evolution [7], resolution of the obtained procedural model is not fixed to the reference image resolution, and our approach does not need any interactive hints on tree branching structure.

In the future research, we would like to do further research on metrics for comparison of rendered and reference images, focusing on topological similarities between the phenotype and a reference image (e.g. number of subbranches and their lengths). Multiple metrics could be used and combined with the use of multi-objective search, and possibly combined with interactive methods for optimization. One of the ideas for future research may also include another encoding aspect of the procedural model using evolution of line segments for vector parameters.

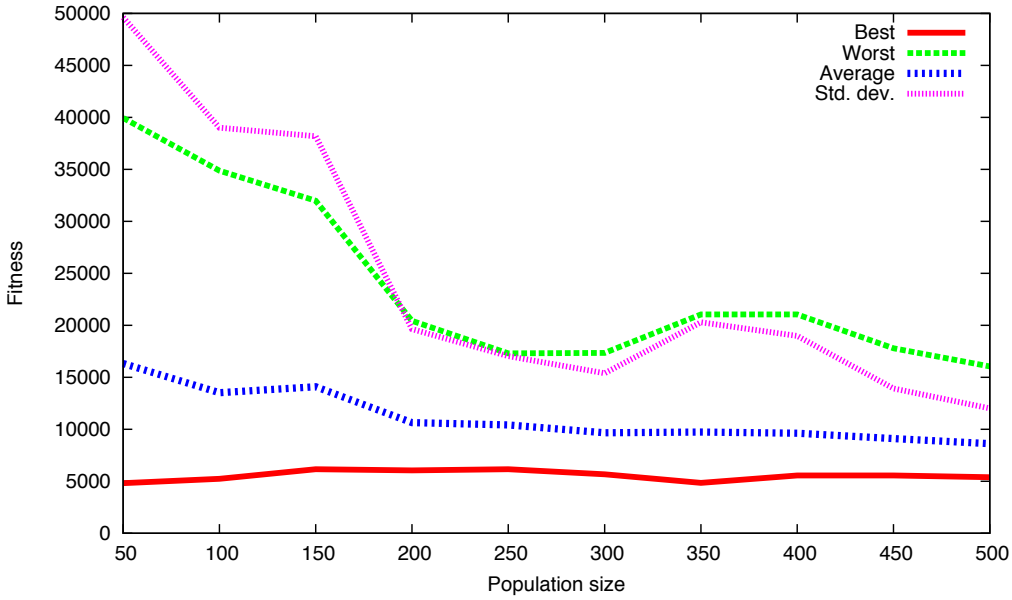

Figure 7: Algorithm DE performance by obtained fitness, dependent on the population size.

References

[1] H.A. Abbass, R. Sarker, C. Newton, PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems, in: Proceedings of the Congress on Evolutionary Computation 2001, volume 2, IEEE Service Center, Piscataway, New Jersey, 2001, pp. 971–978.

[2] B. Alatas, E. Akin, A. Karci, MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules, Applied Soft Computing 8 (2008) 646–656.

[3] M. Ali, Differential evolution with preferential crossover, European Journal of Operational Research 127 (2007) 1137–1147.

[4] M. Aono, T. Kunii, Botanical tree image generation, IEEE Computer Graphics and Applications 4 (1984) 10–34.

[5] B. Bošković and J. Brest and A. Zamuda and S. Greiner and V. Žumer, History Mechanism Supported Differential Evolution for Chess Evaluation Function Tuning, Soft Computing - A Fusion of Foundations,

Methodologies and Applications (In Press). DOI: 10.1007/s00500-010-0593-z.

[6] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, New York, 1996.

[7] D. Beaumont, S. Stepney, Grammatical Evolution of L-systems, in: The 2009 IEEE Congress on Evolutionary Computation CEC 2009, IEEE Press, 2009, pp. 2446–2453.

[8] J. Bloomenthal, Modeling the mighty maple, in: B.A. Barsky (Ed.), SIGGRAPH '85 Conference Proceedings, San Francisco, CA, 22–26 July 1985, pp. 305–311.

[9] J. Brest, Differential Evolution with Self-Adaptation, Encyclopedia of Artificial Intelligence (2009) 488–493.

[10] J. Brest, B. Bošković, S. Greiner, V. Žumer, M.S. Maučec, Performance comparison of self-adaptive and adaptive differential evolution algorithms, Soft Computing - A Fusion of Foundations, Methodologies and Applications 11 (2007) 617–629.

[11] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Žumer, Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems, IEEE Transactions on Evolutionary Computation 10 (2006) 646–657.

[12] J. Brest, M.S. Maučec, Population Size Reduction for the Differential Evolution Algorithm, Applied Intelligence 29 (2008) 228–247.

[13] J. Brest, A. Zamuda, B. Bošković, M.S. Maučec, V. Žumer, Dynamic Optimization using Self-Adaptive Differential Evolution, in: IEEE Congress on Evolutionary Computation 2009, IEEE Press, 2009, pp. 415–422.

[14] J. Brest, A. Zamuda, B. Bošković, M.S. Mauec, V. Žumer, High-dimensional Real-parameter Optimization Using Self-adaptive Differential Evolution Algorithm with Population Size Reduction, in: 2008 IEEE World Congress on Computational Intelligence, IEEE Press, 2008, pp. 2032–2039.

[15] J. Brest, A. Zamuda, B. Bošković, V. Žumer, An Analysis of the Control Parameters Adaptation in DE, in: U.K. Chakraborty (Ed.), Advances in Differential Evolution, Studies in Computational Intelligence, volume 143, Springer, 2008, pp. 89–110.

[16] J. Brest, A. Zamuda, I. Fister, M.S. Maučec, Large Scale Global Optimization using Self-adaptive Differential Evolution Algorithm, in: IEEE World Congress on Computational Intelligence 2010, July 18 - 23, Barcelona, Spain, pp. 3718–3725.

[17] A. Caponio, F. Neri, V. Tirronen, Super-fit control adaptation in memetic differential evolution frameworks, Soft Computing-A Fusion of Foundations, Methodologies and Applications 13 (2009) 811–831.

[18] T.T. Chang, H.C. Chang, An efficient approach for reducing harmonic voltage distortion in distribution systems with active power line conditioners, IEEE Transactions on Power Delivery 15 (2000) 990–995.

[19] X. Chen, B. Neubert, Y.Q. Xu, O. Deussen, S.B. Kang, Sketch-based tree modeling using markov random field, in: SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers, ACM, New York, NY, USA, 2008, pp. 1–9.

[20] C. Darwin, On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life, John Murray, 1859.

[21] S. Das, A. Abraham, U. Chakraborty, A. Konar, Differential Evolution Using a Neighborhood-based Mutation Operator, IEEE Transactions on Evolutionary Computation 13 (2009).

[22] S. Das, P. Suganthan, Differential Evolution: A Survey of the State-of-the-art, IEEE Transactions on Evolutionary Computation 15 (2011) 4–31.

[23] O. Deussen, C. Colditz, M. Stamminger, G. Drettakis, Interactive visualization of complex plant ecosystems, in: Proceedings of the conference on Visualization 2002, pp. 219–226.

[24] A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter Control in Evolutionary Algorithms, IEEE Trans. on Evolutionary Computation 3 (1999) 124–141.

[25] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing (Natural Computing Series), Springer, 2003.

[26] H.Y. Fan, J. Lampinen, A Trigonometric Mutation Operation to Differential Evolution, Journal of Global Optimization 27 (2003) 105–129.

[27] V. Feoktistov, Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[28] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[29] N. Hansen, A. Ostermeier, Completely Derandomized Self-Adaptation in Evolution Strategies, Evolutionary Computation 9 (2001) 159–195.

[30] J. Holland, Adaptation In Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, 1975.

[31] M. Holton, Strands, gravity, and botanical tree imagery, Comput. Graph. Forum 13 (1994) 57–67.

[32] V.L. Huang, A.K. Qin, K. Deb, E. Zitzler, P.N. Suganthan, J.J. Liang, M. Preuss, S. Huband, Problem Definitions for Performance Assessment & Competition on Multi-objective Optimization Algorithms, Technical Report TR-07-01, Nanyang Technological University et. al., Singapore, 2007.

[33] R. Joshi, A. Sanderson, Minimal representation multisensor fusion using differential evolution, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 29 (1999) 1083–4427.

[34] P. Korosec, J. Silc, B. Filipic, The differential ant-stigmergy algorithm, Information Sciences (In Press, DOI: 10.1016/j.ins.2010.05.002, 2011).

[35] A. Lindenmayer, Mathematical models for cellular interactions in development, I & II, J. Theor. Biol. 18 (1968) 280–315.

[36] B. Lintermann, O. Deussen, Interactive modeling of plants, IEEE Computer Graphics and Applications 19 (1999) 56–65.

[37] J. Liu, J. Lampinen, A Fuzzy Adaptive Differential Evolution Algorithm, Soft Comput. 9 (2005) 448–462.

[38] Y. Liu, X. Yao, Q. Zhao, T. Higuchi, Scaling Up Fast Evolutionary Programming with Cooperative Coevolution, in: Proceedings of the 2001 Congress on Evolutionary Computation CEC 2001, IEEE Press, 2001, pp. 1101–1108.

[39] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Applied Soft Computing 11 (2011).

[40] S. von Mammen, C. Jacob, The Evolution of Swarm Grammars: Growing Trees, Crafting Art and Bottom-Up Design, IEEE Computational Intelligence Magazine 4 (2009) 10–19.

[41] U. Maulik, I. Saha, Modified differential evolution based fuzzy clustering for pixel classification in remote sensing imagery, Pattern Recognition 42 (2009) 2135–2149.

[42] M. Mernik, J. Heering, A.M. Sloane, When and how to develop domain-specific languages, ACM Comput. Surv. 37 (2005) 316–344.

[43] E. Mezura-Montes, B.C. Lopez-Ramirez, Comparing bio-inspired algorithms in constrained optimization problems, The 2007 IEEE Congress on Evolutionary Computation (25-28 Sept. 2007) 662–669.

[44] E. Mezura-Montes, J. Velázquez-Reyes, C.A.C. Coello, A comparative study of differential evolution variants for global optimization, in: GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation, ACM Press, New York, NY, USA, 2006, pp. 485–492.

[45] E. Mininno, F. Neri, F. Cupertino, D. Naso, Compact Differential Evolution, IEEE Transactions on Evolutionary Computation 15 (2011) 32–54.

[46] F. Neri, E. Mininno, Memetic compact differential evolution for cartesian robot control, IEEE Computational Intelligence Magazine 5 (2010) 54–65.

[47] F. Neri, V. Tirronen, Recent Advances in Differential Evolution: A Survey and Experimental Analysis, Artificial Intelligence Review 33 (2010) 61–106.

[48] B. Neubert, T. Franken, O. Deussen, Approximate image-based tree-modeling using particle flows, ACM Trans. Graph. 26 (2007) 88.

[49] P.E. Oppenheimer, Real time design and animation of fractal plants and trees, Computer Graphics 20 (1986) 55–64.

[50] S. Papert, D. Watt, A. diSessa, S. Weir, Final Report of the Brookline LOGO Project, Technical Report 545, MIT AI Lab, 1979.

[51] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Natural Computing Series, Springer-Verlag, Berlin, Germany, 2005.

[52] P. Prusinkiewicz, A. Lindenmayer, The Algorithmic Beauty of Plants, Springer-Verlag, 1990.

[53] A. Qin, V. Huang, P. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation 13 (2009) 398–417.

[54] L. Quan, Image-Based Modeling, Springer Publishing Company, Incorporated, 1st edition, 2010.

[55] W. Reeves, Approximate and probabilistic algorithms for shading and rendering structured particle systems, Proceedings of SIGGRAPH'85 (1985) 313–322.

[56] D. Reinhardt, E.R. Pesce, P. Stieger, T. Mandel, K. Baltensperger, M. Bennett, J. Traas, J. Friml, C. Kuhlemeier, Regulation of phyllotaxis by polar auxin transport, Nature (2003) 255–260.

[57] T. Robič, B. Filipič, DEMO: Differential Evolution for Multiobjective Optimization, in: Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization – EMO 2005, volume 3410 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 520–533.

[58] A. Runions, B. Lane, P. Prusinkiewicz, Modeling trees with a space colonization algorithm, Prague, Czech Republic (2007) 63–70.

[59] N. Salvatore, A. Caponio, F. Neri, S. Stasi, G. Cascella, Optimization of Delayed-State Kalman-Filter-Based Algorithm via Differential Evolution for Sensorless Control of Induction Motors, IEEE Transactions on Industrial Electronics 57 (2009) 385–394.

[60] I. Shlyakhter, M. Rozenoer, J. Dorsey, S. Teller, Reconstructing 3D tree models from instrumented photographs, IEEE Computer Graphics and Applications (2001) 53–61.

[61] R. Storn, K. Price, Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization 11 (1997) 341–359.

[62] D. Strnad, N. Guid, Modeling trees with hypertextures, Comput. Graph. Forum 23 (2004) 173–188.

[63] K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.P. Chen, C.M. Chen, Z. Yang, Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization, Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, http://nical.ustc.edu.cn/cec08ss.php, 2007.

[64] J. Teo, Exploring dynamic self-adaptive populations in differential evolution, Soft Computing - A Fusion of Foundations, Methodologies and Applications 10 (2006) 673–686.

[65] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, T. Rossi, An enhanced memetic differential evolution in filter design for defect detection in paper production, Evolutionary Computation 16 (2008) 529–555.

[66] T. Tušar, P. Korošec, G. Papa, B. Filipič, J. Šilc, A comparative study of stochastic optimization methods in electric motor design, Applied Intelligence 2 (2007) 101–111.

[67] J. Tvrdík, Adaptation in differential evolution: A numerical comparison, Applied Soft Computing 9 (2009) 1149–1155.

[68] M. Varadarajan, K. Swarup, Differential evolution approach for optimal reactive power dispatch, Applied Soft Computing 8 (2008) 1549 – 1561.

[69] S. von Mammen, C. Jacob, Genetic Swarm Grammar Programming: Ecological Breeding Like a Gardener, in: D. Srinivasan, L. Wang (Eds.), 2007 IEEE Congress on Evolutionary Computation, IEEE Computational Intelligence Society, IEEE Press, Singapore, 2007, pp. 851–858.

[70] J. Weber, J. Penn, Creation and rendering of realistic trees, Proceedings of SIGGRAPH '95 (1995) 119–128.

[71] X. Yao, Y. Liu, G. Lin, Evolutionary Programming Made Faster, IEEE Transactions on Evolutionary Computation 3 (1999) 82–102.

[72] O. Yogev, A.A. Shapiro, E.K. Antonsson, Computational Evolutionary Embryogeny, IEEE Transactions on Evolutionary Computation 14 (2010) 301–325.

[73] D. Zaharie, Influence of crossover on the behavior of Differential Evolution Algorithms, Applied Soft Computing 9 (2009) 1126–1138.

[74] A. Zamuda, J. Brest, B. Bošković, V. Žumer, Differential Evolution for Multiobjective Optimization with Self Adaptation, in: The 2007 IEEE Congress on Evolutionary Computation CEC 2007, IEEE Press, 2007, pp. 3617–3624.

[75] A. Zamuda, J. Brest, B. Bošković, V. Žumer, Large Scale Global Optimization Using Differential Evolution with Self Adaptation and Cooperative Co-evolution, in: 2008 IEEE World Congress on Computational Intelligence, IEEE Press, 2008, pp. 3719–3726.

[76] A. Zamuda, J. Brest, B. Bošković, V. Žumer, Differential Evolution with Self-adaptation and Local Search for Constrained Multiobjective Optimization, in: IEEE Congress on Evolutionary Computation 2009, IEEE Press, 2009, pp. 195–202.

[77] A. Zamuda, J. Brest, B. Bošković, V. Žumer, Woody Plants Model Recognition by Differential Evolution, in: The Fourth International Conference on Bioinspired Optimization Methods and their Applications, May 20 - 21 2010, Ljubljana, Slovenia, pp. 205–215.

[78] A. Zamuda, J. Brest, N. Guid, V. Žumer, Construction of virtual trees within ecosystems with ecomod tool, in: Proceedings of IPSI-2006 Slovenia, International Conference on Advances in the Internet, Processing, Systems, and Interdisciplinary Research, p. 15.

[79] A. Zamuda, J. Brest, N. Guid, V. Žumer, Modelling, Simulation, and Visualization of Forest Ecosystems, in: The IEEE Region 8 EUROCON 2007: International conference on "Computer as a tool", September 9-12, 2007, Warsaw, Poland, IEEE Press, 2007, pp. 2600–2606.

[80] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, Trans. Evol. Comp 13 (2009) 945–958.

[81] Q. Zhang, A. Zhou, S. Zhao, P.N. Suganthan, W. Liu, S. Tiwari, Multi-objective optimization Test Instances for the CEC 2009 Special Session and Competition, Technical Report CES-487, 2007.

[82] W. Zhu, Nonlinear optimization with a massively parallel evolution strategy - pattern search algorithm on graphics hardware, Applied Soft Computing, 11 (2011).

[83] K. Zielinski, P. Weitkemper, R. Laur, K.D. Kammeyer, Optimization of Power Allocation for Interference Cancellation With Particle Swarm Optimization, IEEE Transactions on Evolutionary Computation 13 (2008) 128–150.