Coevolution Produces an Arms Race Among Virtual Plants

Marc Ebner, Adrian Grigore, Alexander Heffner, and Jürgen Albert

Universität Würzburg, Lehrstuhl für Informatik II
Am Hubland, 97074 Würzburg, Germany
ebner@informatik.uni-wuerzburg.de
http://www2.informatik.uni-wuerzburg.de

Abstract. Creating interesting virtual worlds is a difficult task. We are using a variant of genetic programming to automatically create plants for a virtual environment. The plants are represented as context-free Lindenmayer systems. OpenGL is used to visualize and evaluate the plants. Our plants have to collect virtual sunlight through their leaves in order to reproduce successfully. Thus we have realized an interaction between the plant and its environment. Plants are either evaluated separately or all individuals of a population at the same time. The experiments show that during coevolution plants grow much higher compared to rather bushy plants when plants are evaluated in isolation.

1 Motivation

Creating realistic virtual worlds for a person emerged in a virtual environment is very difficult. Apart from artificial objects such as buildings and cars the virtual world should also contain plants, animals and other people to interact with. We have explored the possibility of evolving virtual plants [8]. Evolving virtual plants instead of manually creating plants opens up the possibility to rapidly create a multitude of different plants. These plants do not necessarily have to exist in the real world. But it is important that, whatever the structures may look like, the user recognizes them as plants.

In our work plants are represented as Lindenmayer-systems or L-systems for short [22]. Prusinkiewicz and Lindenmayer [22] have shown previously how complex, photo-realistically looking plants can be created from a relatively small number of rules. Methods for realistic modeling and rendering of plant ecosystems are described by Deussen et al. [7]. We use an evolutionary algorithm, a variant of genetic programming [17, 18] to automatically generate new populations of plants. Probably one of the first experiments in this area was done by Niklas [20]. Niklas performed an adaptive walk through plant space of branching patterns. An experiment in which different branching patterns compete against each other was also made. Other early experiments were done by Jacob [10–13] who also used a variant of genetic programming to evolve context-free and context-sensitive L-Systems which look like plants. Jacob used a combination of the number of blossoms, the number of leaves and the volume of the plant as a fitness function. Broughton et al. [2] evolved three-dimensional objects similar to Dawkins' Biomorphs [6]. They experimented with two different paradigms, Genetic Programming and L-Systems both of which, when interpreted define a three-dimensional object. Coates et al. [3] extended the experiments and evolved shapes which are adapted to specific constraints, i.e. are able to catch or avoid particles moving in a specific direction. Coevolution was used to evolve objects with an enclosure. Ochoa [21] evolved two-dimensional plant morphologies using L-systems. Kokai et al. [15, 16] evolved L-Systems which describe fractal images or structures. Mock [19] evolved plants for an artificial world where the user took the role of a virtual gardener who could select plants for reproduction. Kim [14] developed a model for the evolution of plant morphology. Plants were grown on a two-dimensional lattice. Hornby and Pollack [9] evolved L-Systems which produce tables and investigated the impact the choice of representation has on the result. Representing the individuals as L-systems produced better results in comparison to a direct encoding.

Our experiments differ from the ones done by Jacob [10–12], Mock [19] and Kokai et al. [15, 16] in that we allow interactions between the plant and its environment. Plants need to catch as much virtual sunlight as possible using their leaves. The amount of sunlight which hits the plant is used to calculate a plant's fitness. We either evaluated each plant individually or we evaluated all individuals of a population at the same time. When all individuals are evaluated at the same time then we also have an interaction between the plants of a population. One plant may place its leaves above the leaves of another plant and thereby use up this sunlight which would have otherwise been received by the plant below. We see that coevolution of plants shapes the plants. Plants grow much higher and try to spread their leaves at the top compared to rather bushy looking plants which occur when plants are evaluated independently.

2 Evolution of artificial plants

We have used deterministic, context-free L-systems as a representation for our plants. A context-free L-system consists of an alphabet V, a starting word ω and a set of rules P [22]. The starting word is defined over the alphabet V: $\omega \in V^+$. The rules are defined as a subset of $V \times V^+$. Each rule $(a, \chi) \in P$ consists of a predecessor a and a successor χ where $\chi \in V^*$. If no successor is defined for a predecessor a then we assume that $a \to a$ belongs to the set of rules P.

A new word is derived from the initial word by replacing all letters of the word by their successors. This process is repeated for a specified number of steps. For the experiments which are described below we have used 5 developmental steps. The major difference between L-systems and the usual Chomsky grammar [4] is that in each step all characters of a word are replaced at the same time. This is supposed to model cell division of multi-cellular organisms. After a word has been derived from the starting word we interpret the letters as commands for a virtual drawing device in three-dimensional space. The symbols are read from left to right.

Table 1. Interpretation of the symbols of our alphabet.

Symbol	Description
`f`	draw a branch segment (cylinder) and move forward
`l`	draw a leaf
`[`	push the current state (transformation matrix) onto the stack
`]`	pop state from stack
`>`	22.5°rotation around x axis
`<`	-22.5°rotation around x axis
`\`	22.5°rotation around y axis
`/`	-22.5°rotation around y axis
`+`	22.5°rotation around z axis
`-`	-22.5°rotation around z axis
`A`, ..., `Z`	no operation

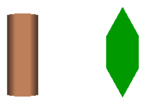


Fig. 1. Building blocks for our plants. A branch segment is shown on the left and a leaf is shown on the right.

We have used a relatively simple alphabet for our experiments. The alphabet consists of the symbols:

$$V = \{\mathtt{f}, \mathtt{l}, \mathtt{+}, \mathtt{-}, \mathtt{<}, \mathtt{>}, \mathtt{/}, \backslash, \mathtt{[}, \mathtt{]}, \mathtt{A}, ..., \mathtt{Z}\}$$

The interpretation of the individual letters is shown in Table 1. The rules for the letters `f` and `A` through `Z` are the only ones which may be changed during the course of the experiment. The other symbols of the alphabet cannot be transformed. Symbols `f` and `l` are used to draw a branch segment and a leaf respectively. Figure 1 shows the building blocks from which our plants are created. All leaves of the plant have the same shape and size. Symbols `+`, `-`, `<`, `>`, `/`, `\` are used to change the orientation of the drawing device. Symbols `[` and `]` can be used to create branching structures. The symbol `[` places the current state (e.g. position and orientation) of the drawing device onto the stack. The symbol `]` pops the topmost state from the stack, thereby restoring the position and orientation of the drawing device to the one which was previously saved. The symbols `A` through `Z` cause no operation and are only used during development.

Each individual consists of one or more rules. The number of rules can be changed by the genetic operators. The predecessor of the first rule is `f`, the predecessor of the second rule is `A`, the predecessor of the third rule is `B` and so on. The initial word from which the plant develops is `f`. Our initial population only contains individuals with the single rule $\mathtt{f} \rightarrow \mathtt{f}$. That is, we start with a population of individuals which only consist of a single branch segment.The fitness of all individuals of the initial population is zero because a branch is not able to collect any sunlight. A typical L-system is shown in Figure 2.

initial word: `A`
rules:

```
f → B/////f
A → [<fCA]/////[<fCA]///////[<fCA]
B → fC
C → [>>l]
```

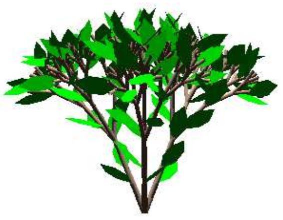


Fig. 2. Sample grammar. The rules are derived from a grammar describing a bush [22].

Plants are evolved using a similar algorithm as in the genetic programming paradigm [17, 18]. To create a new individual for the next generation, we first choose a genetic operator. Each operator is associated with a specific probability that this operator will be chosen. After the type of operator has been determined we either select one or two individuals depending on the type of operator. Crossover operators require two individuals, mutation operators require only a single individual. This process is repeated for a specified number of generations.

- **Permutation**: Two neighboring symbols are exchanged.
- **Mutation**: A randomly selected symbol is replaced with a new symbol.
- **Insertion**: A new symbol is inserted at a random locus.
- **Deletion**: A symbol is deleted at a random locus.
- **One-Point-Crossover**: Crossover is performed by selecting a rule and exchanging all rules between the two individuals which follow the selected rule.
- **Sub-Tree-Crossover**: A randomly selected bracketed subtree is exchanged between two individuals.
- **Add-Branch**: An empty branch is added to an individual.
- **Delete-Branch**: A possibly non-empty bracketed subtree is deleted.
- **Add-Rule** A new rule is appended to the individual, i.e. if the last rule is `C` $\rightarrow \chi$ then `D` → `D` is added.
- **Delete-Rule** The last rule of an individual is deleted.

Fig. 3. Genetic operators.

Figure 3 shows a list of the genetic operators which were used for the experiments. The genetic operators were chosen such that only relatively small steps are possible between successive generations. That is, we did not include operations such as the random generation of subtrees. Reproduction was not included in this set because we found that it was not needed. Whenever an operator cannot be applied the individual is copied unchanged into the next generation, i.e. it is not possible to do a subtree crossover whenever one of the individuals does not contain a bracketed expression.

The fitness of each individual is determined by rendering the plant as an image of size 640×640 using parallel projection viewed from above. Leaves of the plant are drawn in a special color which is unique for each plant and different from the color used to draw the branch segment or the color of the ground. After we have rendered the plant we count the number of pixels which are covered by the plant's leaves. This is a direct measure of the plant's ability to collect sunlight. Using the Z-Buffer to estimate the amount of sunlight hitting a plant was suggested by Beneš [1].

In addition to the number of pixels we also determine the structural complexity of the plant. Branch segments and leaves become more expensive to produce the further they are away from the root of the plant. In our model a branch segment costs 1 point and a leaf costs 3 points. This cost is multiplied with a factor which takes the distance to the root of the plant in account. We define the structural complexity of a plant as

$$\text{complexity} = \sum_{b \in B} \text{cost}_{\text{branch}} \cdot \text{factor}^{\text{height}(b)} + \sum_{l \in L} \text{cost}_{\text{leaf}} \cdot \text{factor}^{\text{height}(l)}$$

where B is the set of branches, L is the set of leaves and height returns the number of branch segments between the current position and the root of the plant. As parameters we have used factor $= 1.1$, $\text{cost}_{\text{branch}} = 1$ and $\text{cost}_{\text{leaf}} = 3$. The fitness of a plant is calculated by subtracting the structural complexity from the amount of light received.

$$\text{fitness} = 10 \cdot \text{points} - \text{complexity}$$

where points is the number of points covered by the plant's leaves. In case of a negative fitness we set fitness to zero. The number of green points is weighted with a factor of 10 which was determined experimentally. A single leaf oriented at a right angle covers 300 points.

3 Experiments

We experimented with a population of 200 individuals with tournament selection and a tournament size of 7. The probability to apply a particular genetic operator was set to 0.1. Two experiments were made. In the first experiment individuals are evaluated in isolation. In this case, the plant is positioned in the center of a square. Leaves which are rendered outside of or below this square do not collect any sunlight. For the second experiment all individuals of the population are evaluated at the same time. Each plant is positioned randomly on the square with a randomly chosen orientation. If a plant places its leaves above another plant's leaves then the one below does not receive as much sunlight as if it were evaluated in isolation. In this case the fitness of a plant also depends on the neighborhood it is growing in. The results of these two experiments are shown in Figure 4 and Figure 5.

Bush-like plants dominate during the first experiment. Plants need to maximize the amount of light received through the leaves while minimizing its structural complexity. Any leaves which are placed below or outside of the square simulating ground only reduce the fitness of the plant. In contrast to the bush-like plants of the first experiment we observed thin and tall plants during the

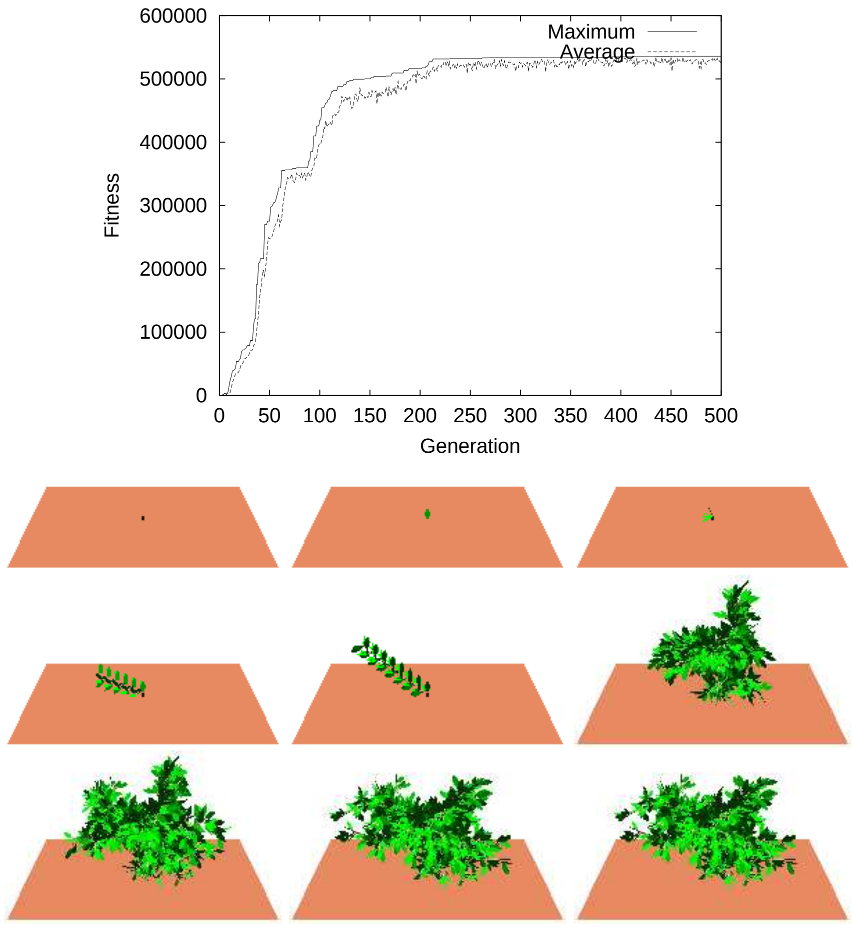
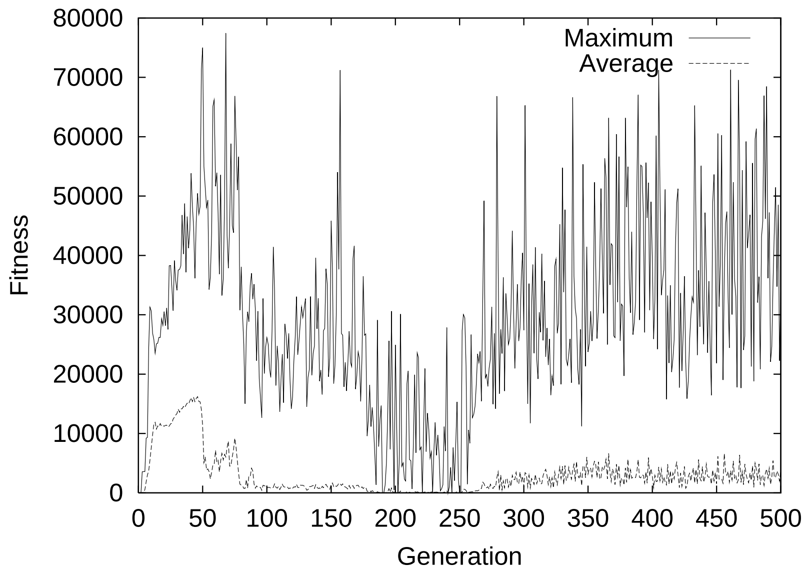


Fig. 4. Results of the experiment when each plant was evaluated in isolation. The graph shows maximum and average fitness values over time. The images show the best individual at generations 0, 4, 8, 16, 32, 64, 128, 256, and 500. Bush-like shapes dominate.

second experiment. A comparison between the best plant which evolved during the two experiments is shown in Figure 6.

During the second experiment average fitness rises initially, then drops very much and the fitness of the best plant starts to oscillate. At this point an evolutionary arms race [5] sets in. This has also been called the red queen hypothesis [23]. According to the red queen hypothesis the fitness of coevolving species may remain at the same level over time. Nevertheless each individual may be continually improving some specific trait. If we look at the two components (amount of light received and structural complexity) of the fitness of an individual we see that the structural complexity of the plant still increases steadily even though



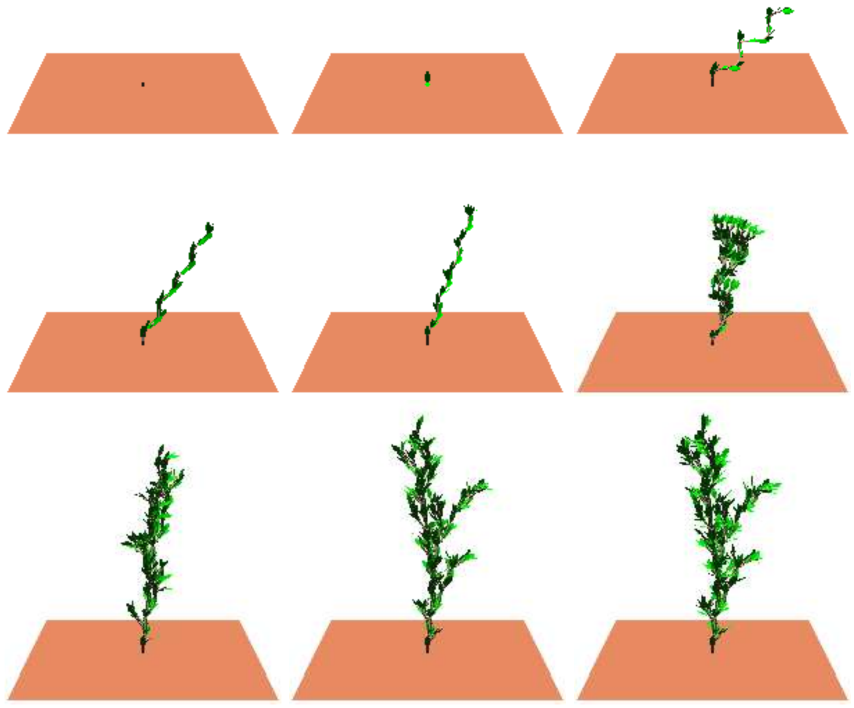


Fig. 5. Results of the experiment when all plants where evaluated at the same time. The graph shows maximum and average fitness values over time. The images show the best individual at generations 0, 4, 8, 16, 32, 64, 128, 256, and 500. The plants grow higher and higher in attempt to gain more sunlight.

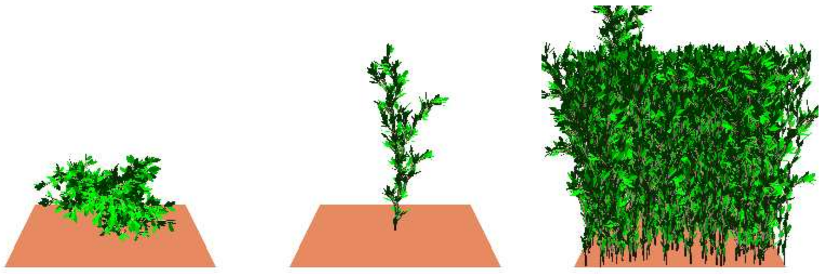


Fig. 6. Plants evolved using coevolution grow higher and are thinner in comparison to plants evolved in isolation. The image on the left shows the best plant when plants were evaluated in isolation, the image in the middle shows the best plant evolved using coevolution. The image on the right shows the whole population of plants.

maximum fitness oscillates heavily and complexity has a negative effect on fitness. Plants need to out-grow their competitors in order to gain more light. If one plant grows higher than another plant it needs more points for its structure but at the same time it also receives more light than its competitors. Figure 8 shows a comparison of the average height and average volume of the plants during the two experiments. Plants evolved during the first experiment occupy a much larger volume than the plants evolved during the second experiment. In comparison, the plants evolved during the second experiment grow higher than the plants evolved during the first experiment.

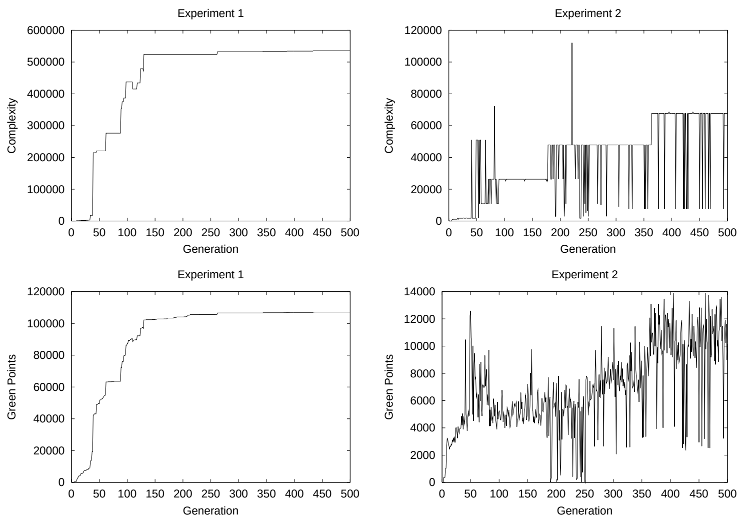


Fig. 7. The graphs on the left show the fitness components for the first experiment. The graphs on the right show the components for the second experiment. The top graphs show the plant's complexity and the bottom graphs show the number of green points.

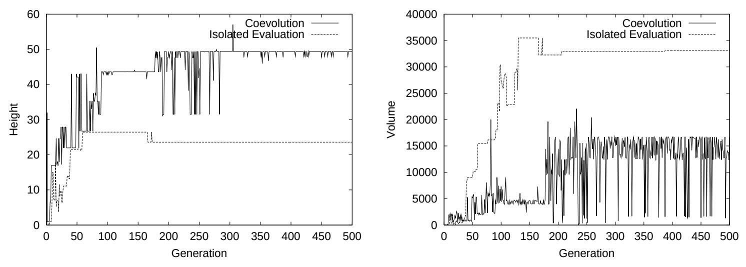


Fig. 8. Comparison of plant height and plant volume for the two experiments.

4 Conclusion

The experiments show how it is possible to use evolutionary techniques to create realistically looking plants for an artificial environment. Plants have to collect virtual sunlight through their leaves. Fitness is calculated as the amount of sunlight received minus the cost for creating the plant. Plants were evaluated either individually or using coevolution. During coevolution we evaluated all plants at the same time. In this case the fitness of a plant depends on its ability to collect sunlight as well as on the neighborhood it is growing in. Thus we have realized an interaction between the plant and its environment.

We found that during coevolution an arms race sets in. Plants grow higher and higher in an attempt to collect more sunlight than its neighbor. Plants which are evaluated in isolation look bushier whereas plants which are evaluated using coevolution look tree-like. The data shows that during coevolution even though fitness stays constant or decreases progress is being made. Plants evolved during coevolution keep increasing their structural complexity in order to catch more light than a neighboring plant. Further experiments could include the simulation of gravity and use of collision detection algorithms. At present, the time required to evaluate the individuals preclude these type of experiments.

References

1. B. Beneš. An efficient estimation of light in simulation of plant development. In R. Boulic and G. Hegron (eds.), *Computer Animation and Simulation 96*, pp. 153–165, Berlin, 1996. Springer-Verlag.
2. T. Broughton, A. Tan, and P. S. Coates. The use of genetic programming in exploring 3d design worlds. In R. Junge (ed.), *CAAD Futures 1997. Proc. of the 7th Int. Conf. on Computer Aides Architectural Design Futures, Munich, Germany*, pp. 885–915, Dordrecht, 1997. Kluwer Academic Publishers.
3. P. Coates, T. Broughton, and H. Jackson. Exploring three-dimensional design worlds using Lindenmeyer systems and genetic programming. In P. J. Bentley (ed.), *Evolutionary Design by Computers*, pp. 323–341. Morgan Kaufmann, 1999.
4. M. D. Davis and E. J. Weyuker. *Computability, Complexity, and Languages*. Academic Press Limited, San Diego, CA, 1983.

5. R. Dawkins and J. R. Krebs. Arms races between and within species. *Proc. R. Soc. Lond. B*, 205:489–511, 1979.
6. R. Dawkins. *The Blind Watchmaker.* W. W. Norton & Company, New York, 1996.
7. O.Deussen, P.Hanrahan, B.Lintermann, R.Měch, M.Pharr, and P.Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH '98 Conf. Proc., Computer Graphics, Orlando, Florida*, pp. 275–286. ACM Press, 1998.
8. A. Grigore, A. Heffner, M. Ebner, and J. Albert. Evolution virtueller Pflanzen. Technical Report 280, Universität Würzburg, Lehrstuhl für Informatik II, Am Hubland, 97074 Würzburg, Germany, August 2001.
9. G. S. Hornby and J. B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proc. of the 2001 Congress on Evolutionary Computation, COEX, Seoul, Korea*, pp. 600–607. IEEE Press, 2001.
10. C. Jacob. Genetic L-system programming. In Y. Davudor, H.-P. Schwefel, and R. Männer (eds.), *Parallel Problem Solving from Nature – PPSN III. The 3rd Int. Conf. on Evolutionary Computation. Jerusalem, Israel*, pp. 334–343, Berlin, 1994. Springer-Verlag.
11. C. Jacob. Evolution programs evolved. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel (eds.), *Parallel Problem Solving from Nature – PPSN IV. The 4th Int. Conf. on Evolutionary Computation. Berlin, Germany*, pp. 42–51, Berlin, 1996. Springer-Verlag.
12. C. Jacob. Evolving evolution programs: Genetic programming and L-systems. In J.R. Koza, D.E. Goldberg, D.B. Fogel, and R.L. Riolo (eds.),*Proc. of the 1st Annual Conf. on Genetic Programming*, pp.107–115,Cambridge,MA,1996. The MIT Press.
13. C. Jacob. Evolution and coevolution of developmental programs. *Computer Physics Communications*, pp. 46–50, 1999.
14. J. T. Kim. LindEvol: Artificial models for natural plant evolution. *Künstliche Intelligenz*, 1:26–32, 2000.
15. G. Kókai, Z. Tóth, and R. Ványi. Application of genetic algorithms with more populations for Lindenmayer systems. In E. Alpaydin and C. Fyfe (eds.), *Int. ICSC Symposium on Engineering of Intelligent Systems EIS'98, University of La Laguna, Tenerife, Spain*, pp. 324–331, Canada/Switzerland, 1998. ICSC Academic Press.
16. G. Kókai, Z. Tóth, and R. Ványi. Evolving artificial trees described by parametric L-systems. In *Proc. of the 1999 IEEE Canadian Conf. on Electrical and Computer Engineering, Shaw Conference Center, Edmonton, Alberta, Canada*, pp. 1722–1727. IEEE Press, 1999.
17. J. R. Koza. *Genetic Programming. On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, MA, 1992.
18. J. R. Koza. *Genetic Programming II. Automatic Discovery of Reusable Programs.* The MIT Press, Cambridge, MA, 1994.
19. K. J. Mock. Wildwood: The evolution of L-system plants for virtual environments. In *Int. Conf. on Evolutionary Computation, Anchorage, AK*, pp. 476–480, 1998.
20. K. J. Niklas. Computer-simulated plant evolution. *Scientific American*, 254(3):68–75, 1986.
21. G. Ochoa. On genetic algorithms and Lindenmayer systems. In *Parallel Problem Solving from Nature - PPSN V*, pp. 335–344, Berlin, 1998. Springer-Verlag.
22. P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants.* Springer Verlag, New York, 1990.
23. L. Van Valen. A new evolutionary law. *Evolutionary Theory*, 1:1–30, July 1973.