

# Coevolution and the Red Queen effect shape virtual plants

Marc Ebner

Received: 30 September 2004 / Revised: 5 December 2005  
© Springer Science + Business Media, LLC 2006

**Abstract** According to the Red Queen hypothesis a population of individuals may be improving some trait even though fitness remains constant. We have tested this hypothesis using a population of virtual plants. The plants have to compete with each other for virtual sunlight. Plants are modeled using Lindenmayer systems and rendered with OpenGL. Reproductive success of a plant depends on the amount of virtual light received as well as on the structural complexity of the plant. We experiment with two different modes of evaluation. In one experiment, plants are evaluated in isolation, while in other experiments plants are evaluated using coevolution. When using coevolution plants have to compete with each other for sunlight inside the same environment. Coevolution produces much thinner and taller plants in comparison to bush-like plants which are obtained when plants are evaluated in isolation. The presence of other individuals leads to an evolutionary arms race. Because plants are evaluated inside the same environment, the leaves of one plant may be shadowed by other plants. In an attempt to gain more sunlight, plants grow higher and higher. The Red Queen effect was observed when individuals of a single population were coevolving.

**Keywords** Red Queen effect · Coevolution · Lindenmayer systems · Artificial plants

## 1. Motivation

One of the main issues of artificial life research is, how to create an environment of continued open ended evolution. Quite often, however, evolution stagnates after some time. For instance, in the field of self reproducing programs, one would think that inoculating the environment with a single self reproducing program would lead to continued open ended evolution. Ray's Tierra system is probably the most well known system of this line of research [40–42]. Ray created an environment where computer programs have to compete

---

**Communicated by:** Una-May O'Reilly

M. Ebner (✉)

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

for CPU cycles. In his experiments, he found that over time, the self reproducing programs became shorter than his handwritten program and an entire ecosystem of virtual organisms emerged. We also experimented with self reproducing programs [10] and found that if the experiments are carried out for a large number of generations, in the end, evolution comes to a halt. In our view, the difficulty with this line of research is, how to get the programs to acquire new capabilities. How can we create such a scenario? In this respect, the study of arms races may lead to new insights on how to create true continued open ended evolution. It is also of interest to find a set of minimal requirements for an arms race to occur [13].

A classic example of an arms race is the coevolution of predator and prey populations such as cheetahs and gazelles. Cheetahs hunt for gazelles while gazelles try to escape from cheetahs on the hunt. If some offspring of the cheetah population is able to run a little faster than other cheetahs, it will be able to catch more gazelles than his competitors. Thus, this trait will probably be reproduced into the next generation. On the other hand, the population of gazelles will become smaller due to the running abilities of the cheetahs. Only those gazelles will survive, which are able to run at least as fast as the cheetahs. At this point, both populations have reached the status quo. Both populations have become faster in the course of time. However, both populations produce the same number of offspring as before. Thus, their fitness has not changed. In this case, both populations have been improving some specific trait (running abilities) even though fitness remained constant. This has been termed the Red Queen hypothesis [45, 50] after the Red Queen chess figure described by Lewis Carroll in “Through the Looking Glass.” The Red Queen noted that one had to do all the running one could do just to stay in the same place. Such pursuit evasion strategies have been evolved in simulation by a number of different researchers using representations such as recurrent networks [3, 4, 15, 32, 35] or genetic programming [27, 44]. Floreano et al. [16] also evolved pursuit evasion strategies for two Khepera robots using recurrent networks.

The Red Queen effect is usually used in the context of two separate populations such as predator and prey. We have visualized the Red Queen effect in a single population of artificial plants [11, 12]. The simulated plants constitute a simple form of ecosystem. According to Ray [43], ecological interactions are an important driving force for evolution. If the environment is sufficiently rich, i.e. the living organisms predominate, then the evolutionary process is primarily concerned with the evolution of adaptations to an ever changing environment. In the experiments which are described below, we will see that the Red Queen effect influences the shape of the plants. Plants grow higher and higher in an attempt to gain more sunlight. Maximum fitness fluctuates around a constant level and sometimes even decreases.

Plants are usually represented as Lindenmayer systems or L-systems for short [39]. Prusinkiewicz and Lindenmayer [39] have created a large variety of complex, photo-realistically looking plants from a relatively small number of rules. Methods for realistic modeling and rendering of plant ecosystems are also described by Deussen et al. [9]. We use an evolutionary algorithm, a variant of genetic programming [27, 29] to automatically generate new generations of plants. One of the first experiments in this area was done by Niklas [34]. Niklas performed an adaptive walk through the space of branching patterns. An experiment in which different branching patterns compete against each other was also made. Koza [28] used genetic programming to discover the rules of an L-system. The method was demonstrated on the evolution of an L-system which generates a Koch curve. McCormack [31] developed an interactive system to evolve parametric L-systems for computer graphics modeling. Other early experiments were done by Jacob [18–22] 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. He also experimented with an ecosystem of different coevolving plant species [22]. Broughton et al. [2] evolved three-dimensional objects similar to Dawkins' Biomorphs [8]. They experimented with two different paradigms, genetic programming and L-systems both of which, when interpreted define a three-dimensional object. Coates et al. [5] 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 [36] evolved two-dimensional plant morphologies using L-systems. Kokai et al. [25, 26] evolved L-systems which describe fractal images or structures. They tried to generate rules which, when executed and viewed, look identical to a given image. Mock [33] evolved plants for an artificial world where the user took the role of a virtual gardener who could select plants for reproduction. He also used an explicit fitness function that rewarded plants which are short but wide. Kim [24] developed a model for the evolution of plant morphology. Plants were grown on a two-dimensional lattice. Hornby and Pollack [17] used L-systems to evolve 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.

Most previous work on artificial plant evolution does not allow interactions between the plant and its environment. The plants in our environment need to catch virtual sunlight through their leaves. They also interact with other individuals from the same population. To estimate how much sunlight is gathered by the plant, we render the plant viewed from the position of the sun. The area covered by the plant's leaves is a measure of the plant's ability to collect sunlight. If a plant doesn't spread its leaves properly, it gathers less sunlight than a plant which exposes all of its leaves to the sun. Also, a plant can shadow itself. The amount of sunlight which hits the leaves of the plant is used to calculate its fitness. We either evaluate each plant individually or we evaluate 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 individual 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 will see that coevolution shapes the plants. If a highly successful plant reproduces more than once into the next generation, the plant will have to compete with one or possibly more copies of itself. Plants grow higher and higher in order to compete for virtual sunlight. If we look at the plant's fitness over time, we will see that although fitness is no longer rising, plants are still evolving. They are adapting to their environment, which in this case is composed of other plants. An evolutionary arms race [7] sets in. In contrast, if we evaluate each plant independently in a separate environment, evolution creates bushy looking plants with increasing fitness.

The article is structured as follows. Evolution of artificial plants is described in Section 2. Calculation of plant fitness is described in Section 3. The set of experiments is described in Section 4 and Section 5 ends with the conclusions.

## 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  $\omega$  and a set of rules  $P$  [39]. 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  $\chi$  where  $\chi \in V^*$ . A single rule is also written as  $a \rightarrow \chi$ . If no successor is defined for a predecessor  $a$ , then we assume that  $a \rightarrow a$  belongs to the set of rules  $P$ .

The major difference between L-systems and the usual Chomsky grammar [6] is that in each step all characters of a word are replaced at the same time. An L-system is basically a string rewrite system where the letters of a word are transformed according to the set of rules. A new word is derived from the initial word by replacing all letters of the word in parallel by their successors. This process is repeated for a specified number of steps. This is supposed to model cell division of multi-cellular organisms [39]. For the experiments which are described below, we have used 5 developmental steps. 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 of the resulting word are read from left to right.

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

$$V = \{f, 1, +, -, <, >, /, \backslash, [, ], A, \dots, Z\}$$

The interpretation of the individual letters is shown in Table 1. The letter *f* produces a branch segment. It draws a cylinder and moves the drawing device forward by a distance equal to the length of the cylinder. The letter *1* draws a leaf at the current position. The position of the drawing device does not change. All leaves of the plant have the same size and shape. Branch segments and leaves are the building blocks from which the plant is created. Symbols *+*, *−*, *<*, *>*, */*, *\* are used to change the orientation of the drawing device. The orientation can be changed in discrete steps of 22.5 degrees in both directions around any of the three axis of the coordinate system, e.g a bent branch segment is created by the sequence *f + f*. Symbols [ and *]* can be used to create branching structures. The symbol [ places the current state (e.g. position

Symbol	Description	
f	draw a branch segment and move forward	
1	draw a leaf	
[	push transformation matrix onto the stack	
]	pop transformation matrix 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	

**Table 1** Interpretation of the symbols of our alphabet.

initial word:  $f$   
rules:

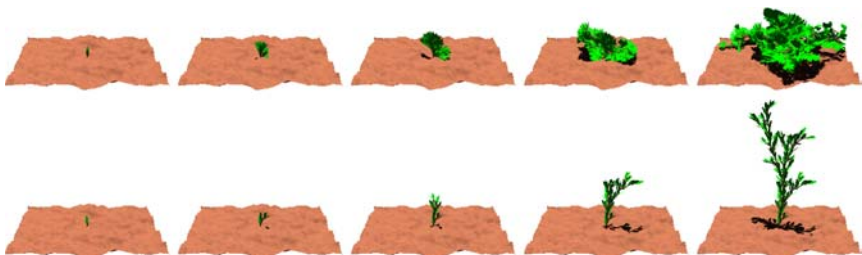
$$\begin{aligned} f &\rightarrow \backslash / f \backslash \backslash < l [A f l] \backslash \backslash - f \\ A &\rightarrow B - f \\ B &\rightarrow C B \\ C &\rightarrow l \end{aligned}$$


**Fig. 1** Sample grammar of an evolved plant. The plant was evolved on a flat landscape using coevolution with a random placement of the offspring

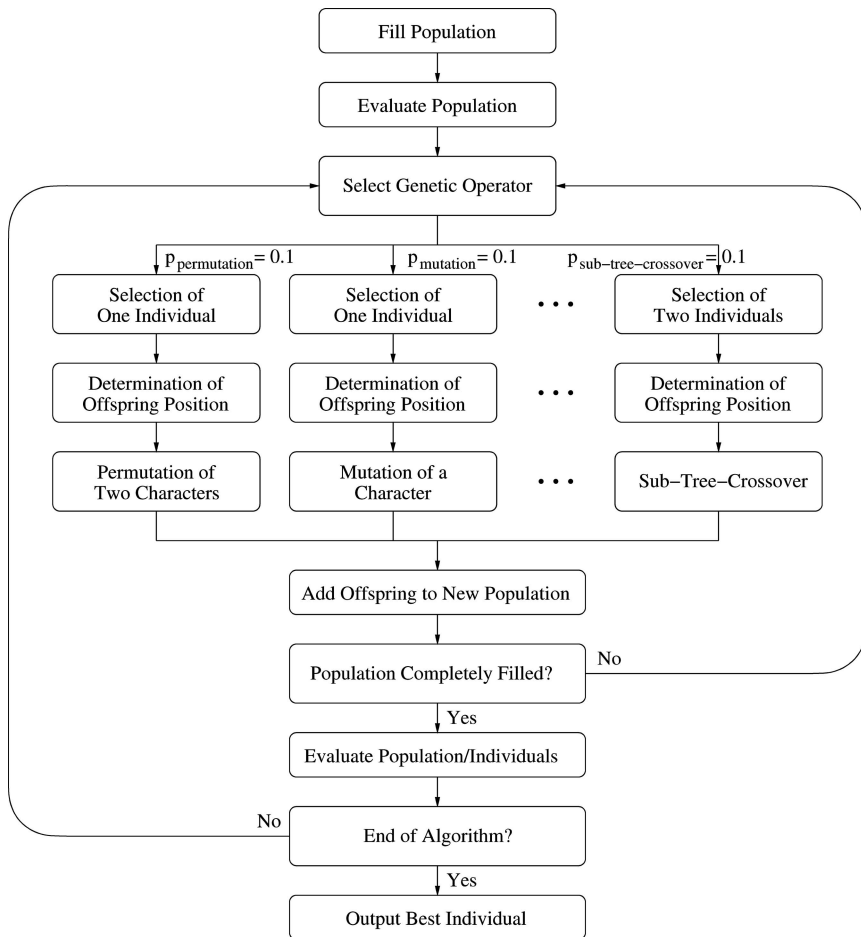
and orientation) of the drawing device onto a stack. The symbol  $]$  pops the topmost state from the stack, thereby restoring the position and orientation of the drawing device to the one which was previously saved. A Y-shaped branching structure is created by the sequence  $f[-f] + f$ . The symbols A through Z cause no operation. They may be used during development to create substructures.

Each individual consists of one or more rules. 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  $f \rightarrow f$ . 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 Fig. 1. A fully developed plant is obtained by applying the rules of the L-system five times to the initial word. This developmental process is shown in Fig. 2.

Plants are evolved using an algorithm similar to the genetic programming paradigm [27, 29]. Figure 3 shows the outline of the evolutionary algorithm. We start by briefly describing the steps of this algorithm. First, the population is initialized and evaluated. All individuals of the first generation are equivalent. They all consist of the single rule  $f \rightarrow f$  with the starting word  $f$ . Each plant has a specific position and orientation inside a two



**Fig. 2** Developmental process of two different plants



**Fig. 3** Outline of evolutionary algorithm. First, the population is initialized and evaluated. Then a genetic operator is chosen. Each genetic operator is selected according to a specific probability. Depending on the type of genetic operator, either one or two parent individuals are selected. Next, the position of the offspring is determined and the selected genetic operator is applied to actually create the offspring. The offspring is then inserted into the new generation. New offspring are created until the population is filled. When the population is completely filled, it is evaluated. The individuals are either evaluated independently or all individuals of the population at the same time. New generations are created until the termination criterion is fulfilled. Finally, the best individual is the result of the algorithm

dimensional area. The initial position of the individuals was chosen depending on the type of experiment. This is described in detail below. To create a new individual for the next generation, we first choose a genetic operator. Each operator is chosen with a specific probability. After the type of operator has been chosen, we select one or two individuals depending on the type of operator. Crossover operators require two individuals, mutation operators require only a single individual. Tournament selection is used to select the parent individuals. This process does not depend on the spatial location of the plants. Offspring are placed somewhere into the vicinity of their parents. Each genetic operator produces either one or two offspring. The offspring are inserted into the next generation. New offspring are created until the population

**Table 2** Genetic operators

Operator	Description
<b>Permutation</b>	Two neighboring symbols of a random successor are exchanged. If a bracket is chosen, then the successor is left unchanged
<b>Mutation</b>	A random character of a random successor is replaced with a new symbol. Brackets cannot be mutated. If no mutable characters exist, then no change is made to the individual
<b>Insertion</b>	A new symbol (excluding brackets) is inserted at a random locus of a random successor
<b>Deletion</b>	A symbol is deleted at a random locus
<b>One-Point-Crossover</b>	A rule is chosen at random and all rules following the selected rule are exchanged between the two individuals
<b>Sub-Tree-Crossover</b>	A bracketed subtree is chosen at random in each individual. Both subtrees are exchanged between the two individuals
<b>Add-Branch</b>	An empty branch is added to the individual. The new branch is placed at a random position of the successor
<b>Delete-Branch</b>	A bracketed subtree is deleted
<b>Add-Rule</b>	A new rule is appended to the individual, i.e. if the last rule is $C \rightarrow \chi$ then $D \rightarrow D$ is added
<b>Delete-Rule</b>	The last rule of an individual is deleted

is filled. Then the population is evaluated. The new population replaces the old population. Thus, the algorithm is a generational algorithm. Note, that it would be more realistic to use a steady state evolutionary algorithm where plant evolution is interleaved with plant development. However, this would have extended the time which is required for the experiments considerably.

Table 2 shows the list of genetic operators which were used for the experiments. Each operator was applied with equal probability. The genetic operators were chosen such that parent and offspring are quite similar. Thus, we only allow relatively small steps from parent to offspring. We did not include operations such as the random generation of subtrees. Also, reproduction was not included in this set. However, if we cannot apply an operator to a selected individual we simply copy the individual into the next generation. For instance, it is not possible to do a subtree crossover whenever one of the individuals does not contain a bracketed expression. In this case, the parent individual is copied unchanged. Some operators change the structure of the rules by rearranging the characters of the rule. Other operators may also change the number of rules used by a single individual. Only the rules for the letters  $f$  and  $A$  through  $Z$  may be changed during the course of the experiment. Rules for symbols such as  $+$  cannot be added. These symbols have the implied rule  $+ \rightarrow +$  which cannot be transformed.

### 3. Plant evaluation

The fitness of a plant is computed from two components: (1) the plant's ability to collect sunlight and (2) the structural complexity of the plant. The first component is computed as



**Fig. 4** The first two images show the same plant viewed from two different positions. The image on the right shows this plant viewed from above. Branch segments are drawn with black and leaves are drawn with green. The number of green pixels in this image is a direct measure of the plant's ability to collect sunlight

follows. It is assumed that the sun is directly above the plant. The plant's ability to collect sunlight depends on the size of leaf area exposed to the sun. We measure this area by rendering the plant using OpenGL [37]. The plant is rendered as an image of size  $512 \times 512$  using parallel projection. The camera is placed above the plant. Thus, we render the plant from the sun's viewpoint. Figure 4 shows such an image. The Z-buffer algorithm of OpenGL ensures that only visible parts of the plant are drawn. A special color which is unique for each plant is used to draw the leaves. This color is also different from the color used to draw the branch segment and also different from the color of the ground. After rendering, we count the number of pixels which have this special color. Since leaves are usually green, we will call this number the number of green pixels. This is a direct measure of the area of the plant's leaves which are exposed to the sun and thus of the plant's ability to collect sunlight.

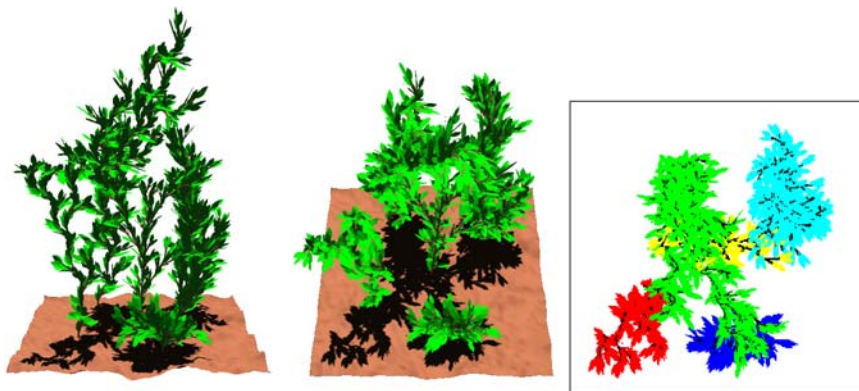
For the first experiment, each individual was evaluated in isolation. The plant was always placed in the center of the environment. For the remaining experiments, each plant has a specific location inside the environment and we only render a single image with all of the individuals. We call this mode of evaluation coevolution. All individuals of the population are evaluated in a single environment. Some researchers use the term coevolution only if two distinct, non-interbreeding populations are used while others do not make this distinction [30]. In our view, the defining essence of coevolution is that the fitness of an individual not only depends on the genotype and the environment, it also depends on the presence of other individuals.

When all individuals of a population are evaluated simultaneously, we use a unique color for each plant to draw the plant's leaves. Therefore, it is easy to determine how much sunlight a plant receives. Figure 5 shows a population of 5 plants. The two images on the left show the population of plants viewed from two different positions. The image on the right shows the image which is used to evaluate the amount of light received per plant. Use of the Z-Buffer to estimate the amount of sunlight hitting a plant was suggested by Beneš [1].

We now address the second component, 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 into account. We define the structural complexity of a plant as

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





**Fig. 5** The two images on the left show 5 different evolved plants. Large plants shadow smaller plants. The image on the right was generated by rendering the plant population from the sun's viewpoint. Each plant has a unique color which is used to draw the plant's leaves. The amount of sunlight received per plant can be obtained easily by counting the number of leaf pixels which belong to each plant

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. The following parameters were used: factor = 1.1. This gives us a cost increase of 10% per height level. The other parameters were set at  $\text{cost}_{\text{branch}} = 1$  and  $\text{cost}_{\text{leaf}} = 3$ . If the cost of a leaf is very low then the entire plant will be covered with leaves. We experimented with different parameters and then settled for the values presented here. The fitness of a plant is calculated by subtracting the structural complexity from the amount of light received,

$$\text{fitness} = \text{gain} \cdot \text{pixels} - \text{complexity}$$

where gain is a constant gain factor and pixels is the number of pixels covered by the plant's leaves. A single leaf oriented at a right angle not covered by anything else covers 196 pixels. In case of a negative fitness we set fitness to zero. The number of green pixels is weighted with a gain factor which was determined experimentally. For the experiments below it was set to 10. If we assume that the number of green pixels is limited, then this factor determines the maximum complexity which can be reached. If the first term is very small, only small plants with low complexity can be created. On the other hand, if the first term is very large, plant complexity can become much larger.

#### 4. Experiments

A set of four experiments was performed. A complete list of parameters for these four experiments is shown in Table 3. We experimented with a population of 200 individuals with tournament selection and a tournament size of 7. All individuals of the first generation have a fitness of zero because they all consist of the single rule  $f \rightarrow f$  with the starting word  $f$ . The probability to apply a particular genetic operator was set to 0.1. The first experiment took over 10 days to complete 500 generations. The other three experiments took between 3 and 4 days to complete. Therefore, we only report on the results of single runs. The data

**Table 3** Parameters which were used for the experiments

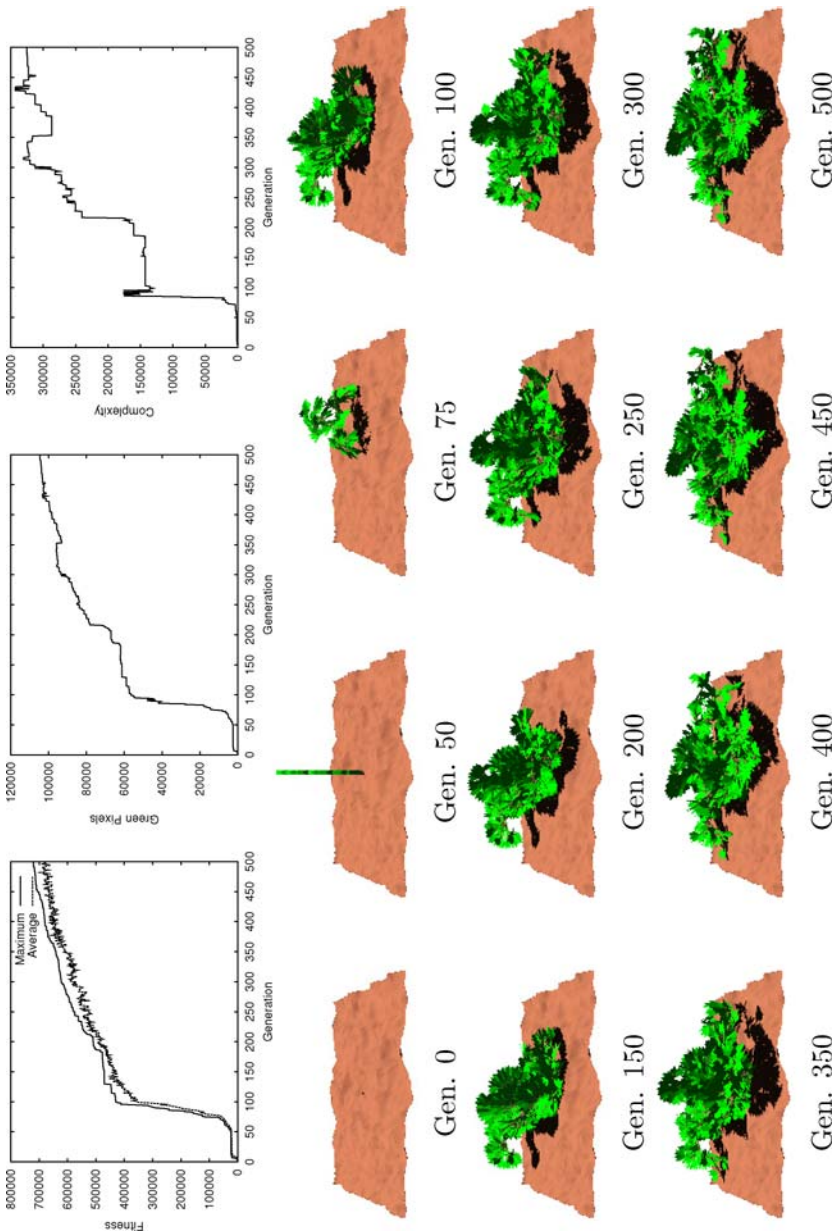
Experiment no.	1	2	3	4
Number of generations	500	500	500	500
Size of population	200	200	200	200
Method of selection	Tournament	Tournament	Tournament	Tournament
Size of tournament	7	7	7	7
Shape of ground factor	Flat	Flat	Sloped	2 Height Levels
$\text{cost}_{\text{branch}}$	1.1	1.1	1.1	1.1
$\text{cost}_{\text{leaf}}$	1.0	1.0	1.0	1.0
gain	3.0	3.0	3.0	3.0
Position of first pop.	10.0	10.0	10.0	10.0
Placement of offspring	Center	Around center	At right edge	At Corner
Coevolution	Center	Close to parent	Close to parent	Close to parent
$P_{\text{Add}} - \text{Rule}$	no	yes	yes	yes
$P_{\text{Delete}} - \text{Rule}$	0.1	0.1	0.1	0.1
$P_{\text{Permutation}}$	0.1	0.1	0.1	0.1
$P_{\text{Mutation}}$	0.1	0.1	0.1	0.1
$P_{\text{Insertion}}$	0.1	0.1	0.1	0.1
$P_{\text{Deletion}}$	0.1	0.1	0.1	0.1
$P_{\text{One-Point-Crossover}}$	0.1	0.1	0.1	0.1
$P_{\text{Sub-Tree-Crossover}}$	0.1	0.1	0.1	0.1
$P_{\text{Add-Branch}}$	0.1	0.1	0.1	0.1
$P_{\text{Delete-Branch}}$	0.1	0.1	0.1	0.1

is not averaged. Although we report only the results for single runs, we observed similar results in earlier experiments [12].

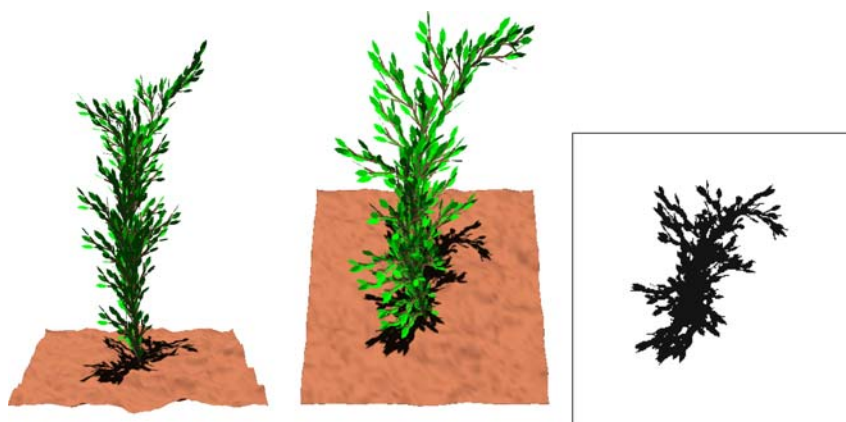
For the first experiment, individuals are evaluated in isolation. In this case, the plant is positioned in the center of a square landscape. This landscape was generated using Perlin noise [38]. Leaves which are rendered outside of or below this landscape do not collect any sunlight. The results of this experiment are shown in Fig. 6. Plants quickly evolve into a bush-like shape. Since a cost is assigned to each leaf and to each branch segment, it is not advantageous for the plant to place part of its structure below the surface. Plants which don't show this behavior are fitter than the plants that do.

For the remaining three experiments, we have used coevolution. Each plant has a particular position and orientation in space. Instead of evaluating the fitness of a plant one after the other, with the plant placed in the middle of the environment, we evaluate all individuals of the population at the same time. For this mode of evaluation we render a single image of the whole population of plants. Each plant is assigned a unique color to draw the plant's leaves. This allows us to determine the amount of light received per plant. 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. Note, that this type of coevolution is different from the usual setup where two different populations are used, e.g. in a predator prey scenario. We work with a single population where an individual of the population has to compete with all other individuals of the same population.

Since each plant has a position and orientation in space, we need a method to determine where a plant's offspring will be placed. The position and orientation of the individuals in the first generation was chosen such that all individuals are located in a random position inside a



**Fig. 6** Experiment 1: Isolated evaluation. The first graph shows maximum and average fitness over time. The second graph shows how the first fitness component (number of green pixels) changes over time. The third graph shows the second fitness component (plant complexity). The images below show the best individual from generation 0, 50, 75, 100, 150, 200, 250, 300, 350, 400, 450, and 500



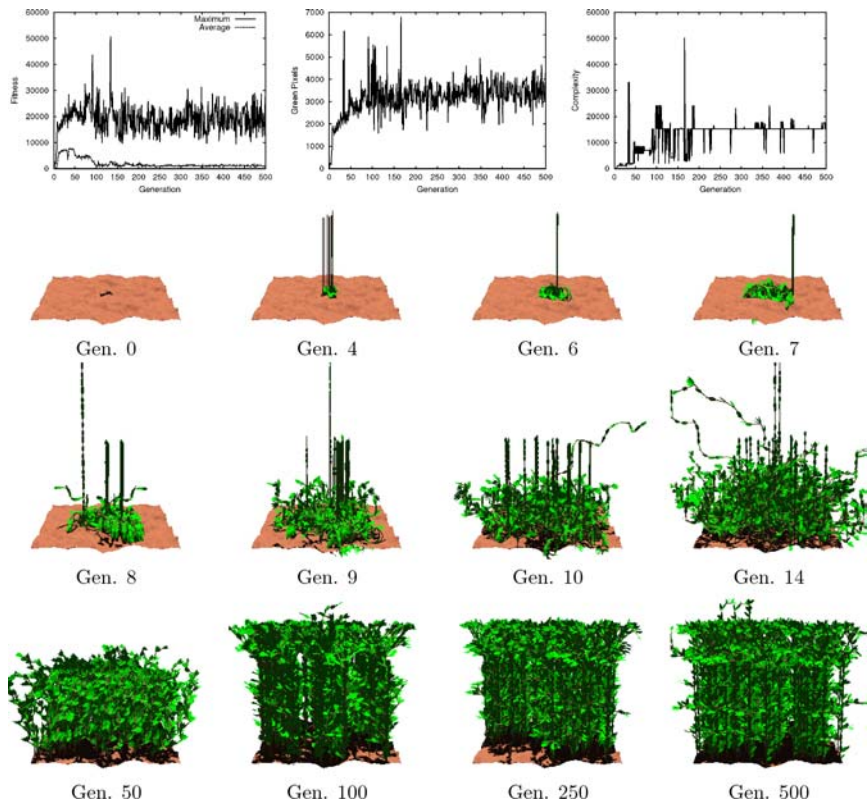
**Fig. 7** The image on the right shows the footprint of the plant which is shown on the left. This image was generated by rendering the plant from the sun's viewpoint. All plant components are drawn in black. Offspring may be placed anywhere on the footprint of the plant. Whenever an offspring is generated, one of the possible locations is chosen at random

small circular area of the environment. Orientation was set randomly. When an offspring is created, it is always placed in the vicinity of the parent. This is done by rendering the plant viewed from above. The plant's offspring is then placed on a random location of the plant's footprint (Fig. 7).

Three different landscapes were used: a flat landscape as in the first experiment, a landscape which has a large vertical slope and a landscape with two different height levels. Again, Perlin noise was used to provide small scale variations of the landscape. Results are shown in Figs. 8, 9 and 10. The first graph shows maximum and average fitness over time. The second graph shows the number of green pixels, the third graph shows plant complexity for the best plant of each generation. The images below the graphs show snapshots of the population for different generations of the evolutionary algorithm. For the experiment on the flat landscape, individuals of the first generation were placed in the center of the environment. From there, the population quickly spread and eventually populated the whole environment. For the sloped landscape, individuals were originally placed on the right side of the environment. Again, individuals quickly spread and populated the available area. For the landscape with two height levels, we placed the individuals in a small area located at the front right corner. Again the population spread and populated both height levels.

The first experiment resulted in bush-like plants. Here, a single plant is most successful if it covers all of the available space while minimizing its structural complexity. It does not pay for the plant to grow very high as this adds to the plant's complexity. This is different for the other three experiments. Here, plants face competition from other individuals. This leads to an arms race where plants grow higher and higher in an attempt to gain more sunlight. Thin and tall plants dominate. Plants effectively try to maximize the number of green pixels, i.e. their ability to gather sunlight. By increasing the number of green pixels, the structural complexity may also be increased without reducing the fitness to zero. If all plants are evaluated inside the same environment, the competition for sunlight results in increasingly larger and larger plants. A comparison of the best plants which were evolved during experiments 1 through 4 is shown in Fig. 11.

If we look at maximum fitness, we see that fitness starts to fluctuate around a constant level very early during the run (approximately at generation 50). However, at this point the

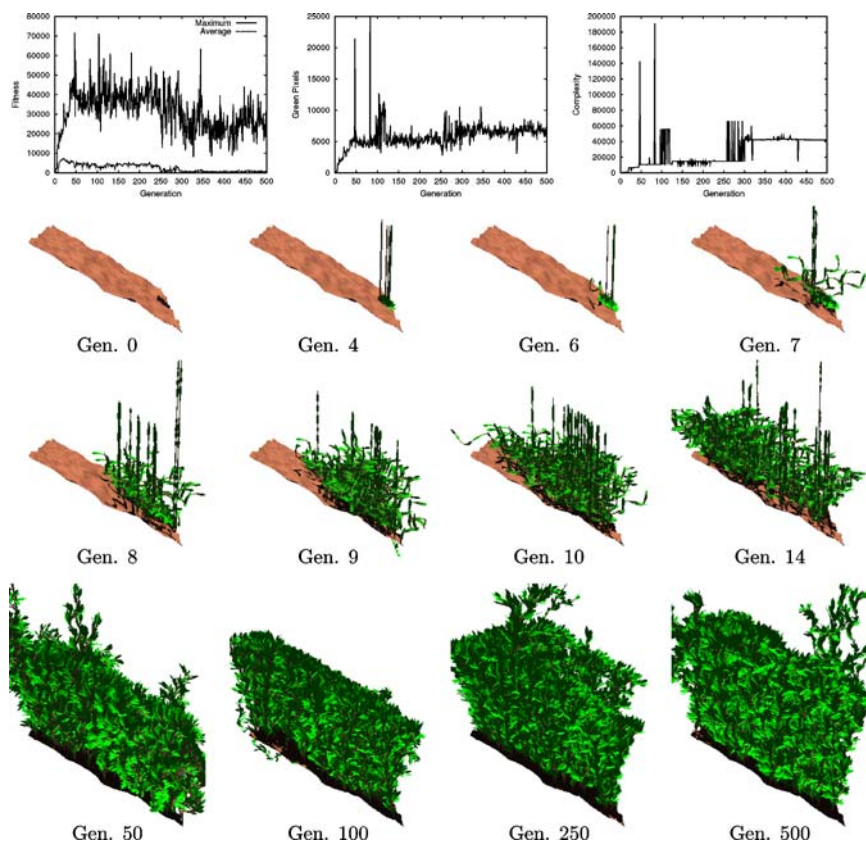


**Fig. 8** Experiment 2: Coevolution in a flat environment. In contrast to experiment 1 the individuals of a population are all evaluated at the same time. Initially, the population is concentrated in a small area in the middle of the environment. The plants soon spread to populate the whole area. From this point onwards an arms race sets in. The images show the population at generation 0, 4, 6, 7, 8, 9, 10, 14, 50, 100, 250, and 500. Note that the plants still evolve after generation 50. Apart from the oscillations, maximum fitness stays largely constant during the remainder of the experiment

evolutionary process has not come to a halt yet. Plants are still evolving. They adapt their shapes to their environment. The structural complexity and the height of the plants is still increasing. Figure 12 shows the height of the best plant for each generation. We can see that apart from the first experiment, plant height increases even though plant height increases the structural complexity and therefore has a negative impact on fitness. Note that the maximum height of the plants was not limited explicitly. However, the number of green pixels per plant is fixed when distributed evenly among all plants. Eventually, plants evolve to the point where no further increase in structural complexity is advantageous. Structural complexity cannot become larger than the first term of the fitness function which depends on the number of green pixels. This also limits the maximum height attainable by the plants.

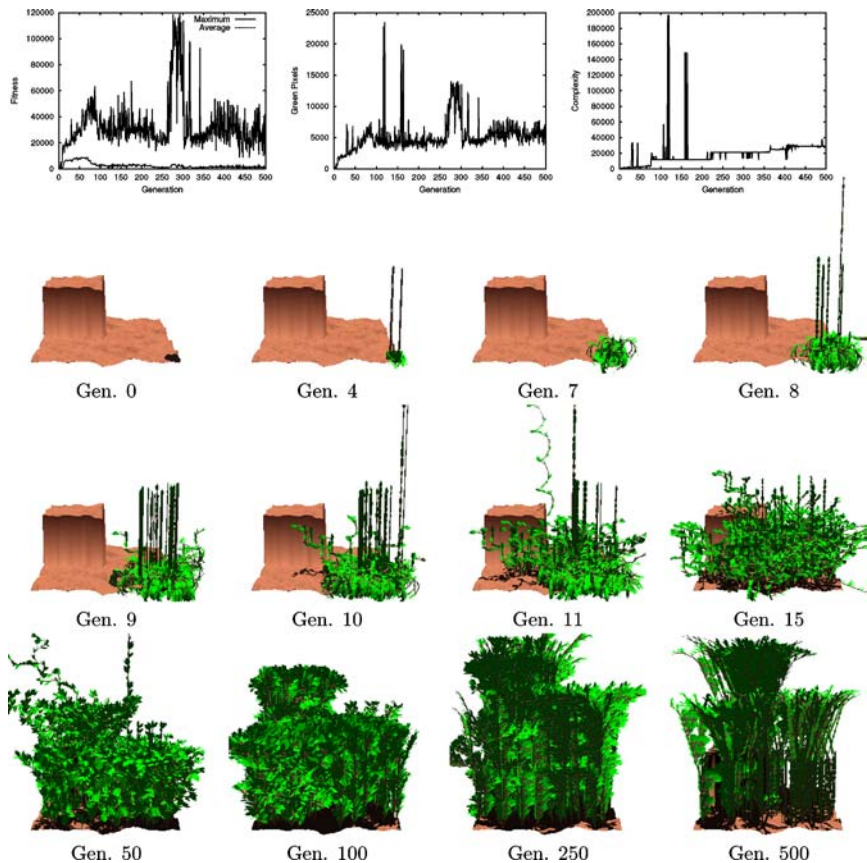
A number of researchers have investigated how evolutionary progress can be measured [3, 30]. Some have suggested adding an external memory of individuals to guarantee progress [14, 23]. Cliff and Miller [3] suggested to perform tests between the best individual of the current generation and the best individual of all previous generations. The result of these tests is entered into a CIAO (Current Individual vs. Ancestral Opponents) graph. Floreano





**Fig. 9** Experiment 3: Coevolution in a sloped environment. The setup is similar to experiment 2 except that a sloped environment was used and the population is initially placed at the right side of the environment. The images show the population at generation 0, 4, 6, 7, 8, 9, 10, 14, 50, 100, 250, and 500. Again, plants still evolve after generation 50. However, maximum fitness decreases after generation 50

and Nolfi [15] suggested to perform a master tournament where the best individual of each generation is compared to the best individual of all other generations. Stanley and Miikkulainen [48] suggested the dominance tournament method to monitor progress in coevolutionary experiments. The dominance tournament method is used here as the number of evaluations is considerably lower when compared to the other methods. The dominance tournament builds a list of dominant strategies. A dominant strategy is a strategy which defeats all previous dominant strategies. The first dominant strategy is the best individual from the first generation. A dominant strategy is added whenever it beats all other dominant strategies which have been found so far. We determine if strategy A beats strategy B by filling half of the population with individuals using strategy A and the second half of the population with individuals using strategy B. The individuals of the population are distributed randomly throughout the environment. The entire population is then evaluated and average fitness of strategies A and B are computed. Strategy B beats strategy A if average fitness of B is significantly higher (at the 1% level). Figures 13, 14 and 15 show the dominant individuals for experiments 2 through 4.

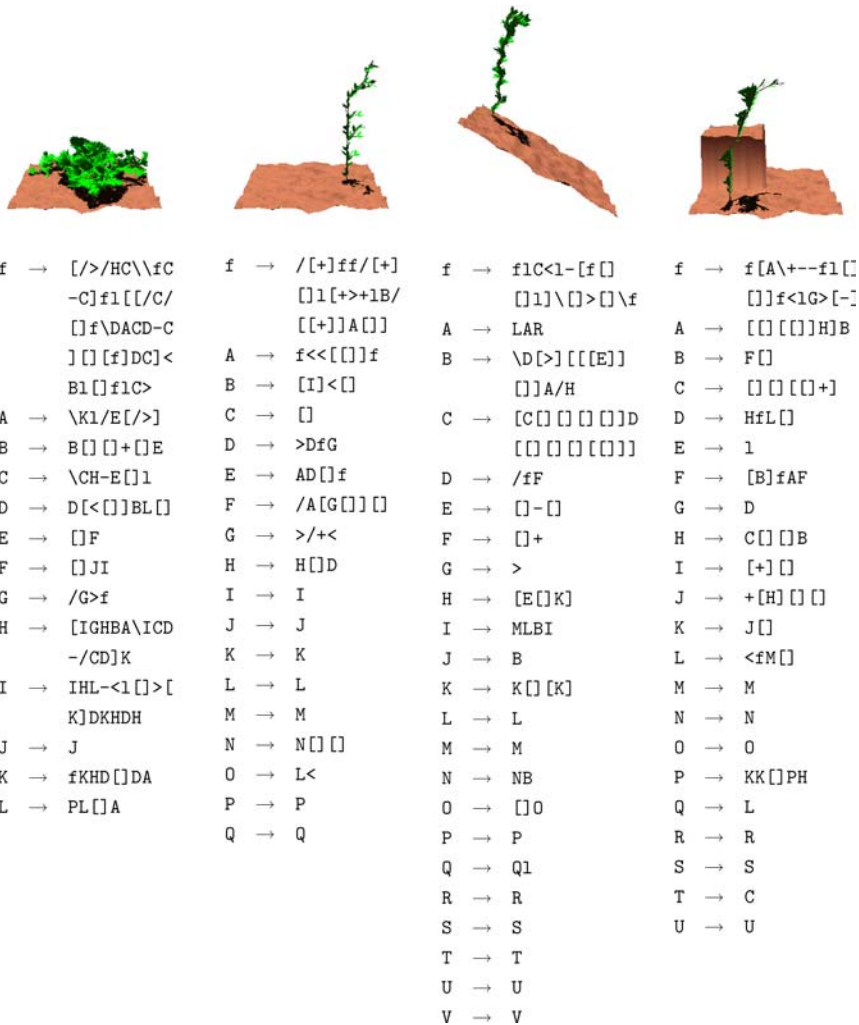


**Fig. 10** Experiment 4: Coevolution in an environment with two different height levels. The populations are initially placed inside a small area in the lower right corner. Over time, it populates both height levels. The images show the population at generation 0, 4, 7, 8, 9, 10, 11, 15, 50, 100, 250, and 500. The population spreads from a small area in the lower right corner and eventually populates the whole area. Again, we see that evolution still continues after the first peak at around generation 75

Let's have a theoretical look on the possible outcomes of this experiment. Stenseth and Maynard Smith [49] note that an ecosystem in a physically constant environment may be in one of two states: (1) a steady state of evolutionary change or (2) evolutionary stasis. If we take the available area (number of pixels in the image) and divide it by the number of plants in the population, we obtain the maximum number of green pixels per plant which can be covered. Let this number be  $m$ .

$$m = \frac{\text{total number of pixels}}{\text{number of individuals in population}}$$

Covering  $m$  pixels on average with a minimal structural complexity is the evolutionary stable strategy for this type of experiment. An evolutionary stable strategy is a strategy which is defined as follows [47]. Let  $A$  and  $B$  be two different strategies and let  $E_A(B)$  be the expected payoff to  $B$  when matched against  $A$ . The strategy  $A$  is an evolutionary stable strategy if for all other strategies  $B$ ,  $E_A(A) > E_A(B)$ , i.e.  $A$  is the best strategy when matched against any other strategy. If for any strategy  $B$ ,  $E_A(A) = E_A(B)$ , i.e. there is some other strategy that does equally well against strategy  $A$ , then it is required that  $E_B(A) > E_B(B)$ , for  $A$  to be an

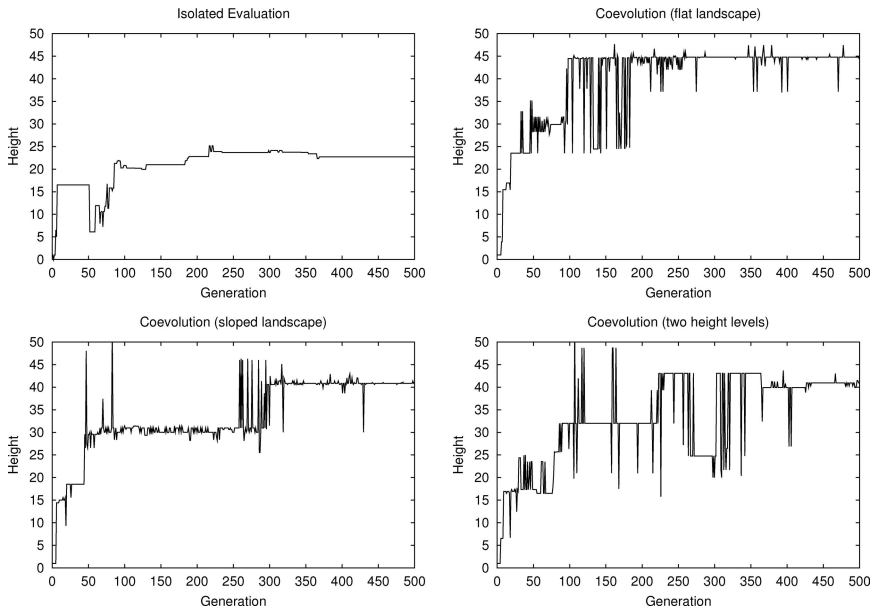


**Fig. 11** Plants evolved using coevolution grew higher and are thinner in comparison to plants which were evaluated in isolation. The image on the top left shows the best plant when plants were evaluated in isolation. The other three images show plants which were evolved using coevolution. The second plant was evolved with a flat environment, the third one with a sloped environment and the third one with an environment with two height levels. The evolved L-system is shown below each image

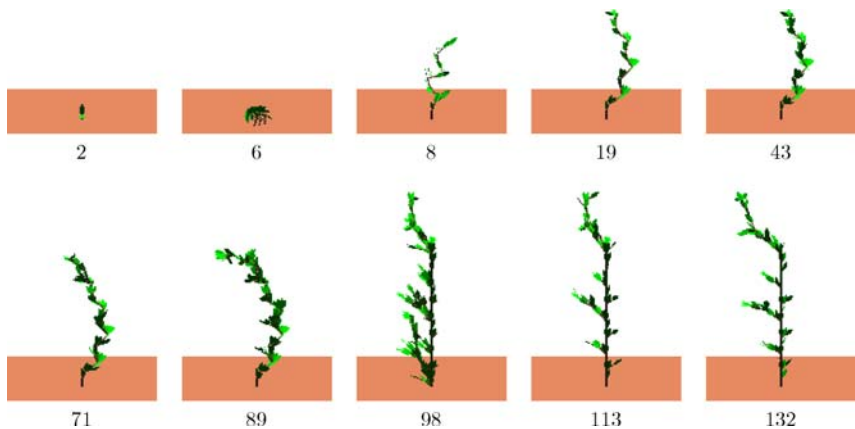
evolutionary stable strategy. In other words, once strategy *A* occurs in the population, no other strategy can overtake the population.

Also see Rosenzweig et al. [46] for an in depth discussion of the Red Queen effect and evolutionary stable strategies. Rosenzweig et al. note that the Red Queen effect appears if there exists traits which are unbounded and which are best at their most extreme values. If traits are constrained or correlated with other traits an evolutionary stable strategy emerges. This is exactly the behavior seen in the experiments we conducted. We first see the Red Queen effect until plant complexity has reached the maximum value which is implicitly set by the total available area and the number of plants in the population.



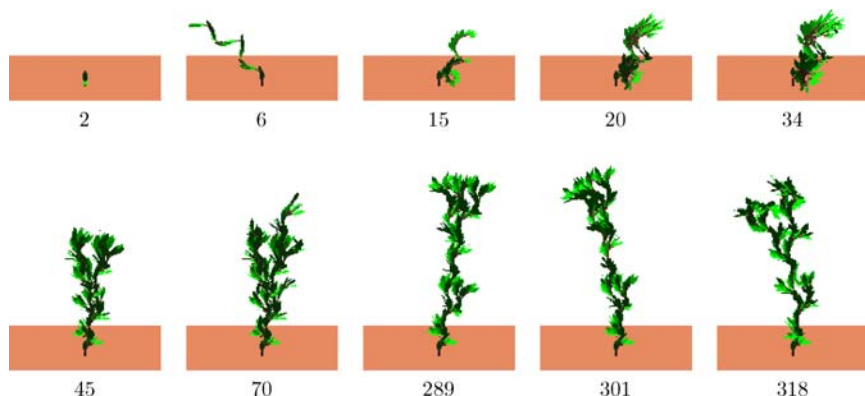


**Fig. 12** Plant height of the best plant for each generation of the four experiment. Isolated evaluation lead to comparatively small plants. For the other three experiments, coevolution was used. Even though fitness remained constant and sometimes even decreased, plant height was increasing over time

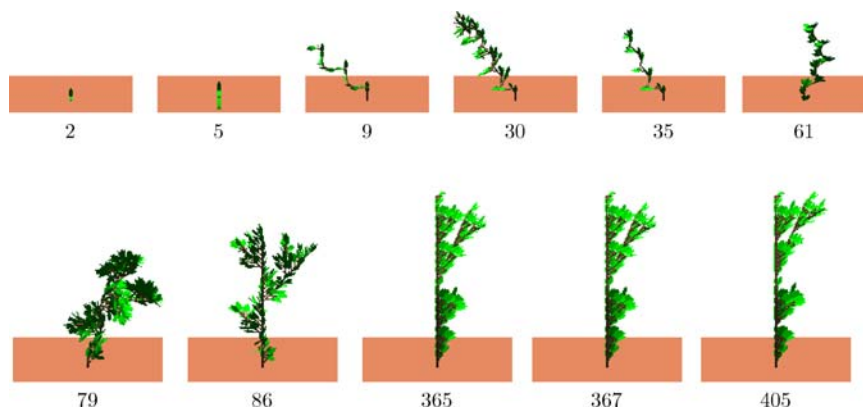


**Fig. 13** Result of dominance tournament for experiment 2. The number below each plant indicates the generation at which this dominant individual was found

If a new mutation causes a plant to grow very high and wide while the remainder of the population is still small, such that it covers more than  $m$  pixels, then this plant will reproduce quite often. This plant will then have to compete with multiple copies of itself in the next generation. Since the orientation of the plants is random, eventually, only  $m$  pixels will be covered. This sets the scene for other plants which also cover  $m$  pixels but don't grow as wide, i.e. they have a smaller structural complexity. So, on average only  $m$  pixels will be covered and plant width will be limited due to the structural complexity factor. Plant height is initially



**Fig. 14** Result of dominance tournament for experiment 3



**Fig. 15** Result of dominance tournament for experiment 4. The phenotypes of individuals from generation 365 and 367 are actually identical

unlimited. However, since maximum number of green pixels per plant is limited, it also limits the maximum height of the plants. Therefore, eventually an evolutionary stable strategy will be reached. A similar situation is reached in other arms races, e.g. the population of cheetahs and gazelles. Maximum speed cannot increase indefinitely due to physical constraints.

## 5. Conclusion

A population of individuals may also adapt to its environment even if fitness remains constant. We have shown this Red Queen effect for a population of artificial plants. Maximum fitness of the population fluctuates around a constant level and sometimes even decreases, yet the plants adapt to their environment. We have modeled the plants using Lindenmayer systems. OpenGL was used to render the plants. The plants have to collect virtual sunlight through their leaves in order to reproduce. 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. 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 which in this case consists of other individuals. If an offspring is generated, it is placed in the vicinity of the plant. The possible locations for the offspring are determined by the plant's footprint. If a highly successful plant creates two or more offspring, then these offspring compete against each other.

The Red Queen effect was visualized for three different environments, an essentially flat landscape, a landscape with a large slope and a landscape with two height levels. The original population of plants was placed in a small area. From this area, plants quickly spread over the available space. After the environment was populated, an arms race set in. 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. Even though structural complexity has a negative impact on fitness, plants grew higher and higher in an attempt to collect more sunlight than their competitors. Plants which were evaluated in isolation are short and look bush-like whereas plants which were evaluated using coevolution look tree-like. Thus, coevolution and the Red Queen effect shaped the virtual plants.

In the end, however, the plants also evolved to an evolutionary stable strategy with no further progress. So the main issue still remains unsolved. How do we create an environment which allows true open ended evolution? It may also be of interest if an environment can be found which leads to a speciation of the plants, i.e. to have one set of plants located in one area and a different set of plants located in another area. Another possible extension would be to see what happens if the developmental and the evolutionary process are interleaved.

In our research we have shown the Red Queen effect in a population of coevolving plants. With a better understanding of the Red Queen effect we may be able to achieve open ended evolution which is capable of continued innovation.

## References

- 1 B. Beneš, "An efficient estimation of light in simulation of plant development," in *Computer Animation and Simulation 96*, R. Boulic and G. Hegron (eds.), Springer-Verlag, Berlin, 1996, pp. 153–165.
- 2 T. Broughton, A. Tan and P. S. Coates, "The use of genetic programming in exploring 3D design worlds," in *CAAD futures 1997. Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures*, R. Junge (ed.), Munich, Germany, Kluwer Academic Publishers, Dordrecht, 4–6, 1997, pp. 885–915.
- 3 D. Cliff and G. F. Miller, "Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations," in *Third European Conference on Artificial Life*, F. Morán, A. Moreno, J. J. Merelo, and P. Chacón (eds.), Springer-Verlag, Berlin, 1995, pp. 200–218.
- 4 D. Cliff and G. F. Miller, "Co-evolution of pursuit and evasion ii: Simulation methods and results," in *From Animals to Animats 4: Proc. of the Fourth International Conference on Simulation of Adaptive Behavior*, P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson (eds.), The MIT Press, Cambridge, Massachusetts, 1996, pp. 506–515.
- 5 P. Coates, T. Broughton and H. Jackson, "Exploring three-dimensional design worlds using Lindenmeyer systems and genetic programming," in *Evolutionary Design by Computers*, P. J. Bentley (ed.), Morgan Kaufmann, 1999, pp. 323–341.
- 6 M. D. Davis and E. J. Weyuker, *Computability, Complexity, and Languages*, Academic Press Limited, San Diego, California, 1983.
- 7 R. Dawkins and J. R. Krebs, "Arms races between and within species," *Proc. R. Soc. Lond. B*, vol. 205; pp. 489–511, 1979.
- 8 R. Dawkins, *The Blind Watchmaker*, W. W. Norton & Company, New York, 1996.
- 9 O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr and P. Prusinkiewicz, "Realistic modeling and rendering of plant ecosystems," in *SIGGRAPH'98 Conference Proc. Computer Graphics*, ACM Press, Orlando, Florida, 1998, pp. 275–286.

- 10 M. Ebner, "A three-dimensional environment for self-reproducing programs," in *Proc. of the 6th European Conference on Artificial Life, ECAL 2001*, J. Kelemen and P. Sosik (eds.), Prague, Czech Republic, Springer-Verlag, Berlin, 2001, pp. 306–315.
- 11 M. Ebner, "Evolution and growth of virtual plants," in *Advances in Artificial Life — Proc. of the 7th European Conference on Artificial Life (ECAL)*, Dortmund, W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler (eds.), Springer-Verlag, Berlin, Germany, 2003, pp. 228–237.
- 12 M. Ebner, A. Grigore, A. Heffner and J. Albert, "Coevolution produces an arms race among virtual plants," in *Genetic Programming: Proc. of the Fifth European Conference, EuroGP 2002, Kinsale, Ireland*, 3–5, J. A. Foster, E. Lutton, J. Miller, C. Ryan and A. G. B. Tettamanzi (eds.), Springer-Verlag, Berlin, 2002, pp. 316–325.
- 13 M. Ebner, R. A. Watson and J. Alexander, "Co-evolutionary dynamics on a deformable landscape," in *Proc. of the 2000 Congress on Evolutionary Computation*, IEEE Press, San Diego Marriott Hotel, La Jolla, CA, 2000, vol. 2, pp. 1284–1291.
- 14 S. G. Ficici and J. B. Pollack, "A game-theoretic memory mechanism for coevolution," in *Proc. of the Genetic and Evolutionary Computation Conference GECCO-2003*, E. Canté-Paz et al. (ed.), Springer-Verlag, Berlin, 2003, pp. 286–297.
- 15 D. Floreano and S. Nolfi, "God save the Red Queen competition in co-evolutionary robotics," in *Genetic Programming 1997: Proc. of the Second International Conference on Genetic Programming*, J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, (eds.), Morgan Kaufmann Publishers, San Francisco, California, 13–16, 1997, pp. 398–406.
- 16 D. Floreano, S. Nolfi and F. Mondada, "Competitive co-evolutionary robotics: From theory to practice," in *From Animals to Animats 5: Proc. of the Fifth International Conference on Simulation of Adaptive Behavior*, R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson (eds.), The MIT Press, 1998, pp. 515–524.
- 17 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*, IEEE Press, COEX, Seoul, Korea, 2001, pp. 600–607.
- 18 C. Jacob, "Genetic L-system programming," in *Parallel Problem Solving from Nature — PPSN III. The Third International Conference on Evolutionary Computation*, Y. Davidor, H.-P. Schwefel, and R. Männer, (eds.), Jerusalem, Israel, Springer-Verlag, Berlin, 1994, pp. 334–343.
- 19 C. Jacob, "Evolution programs evolved," in *Parallel Problem Solving from Nature — PPSN IV. The Fourth International Conference on Evolutionary Computation*, H.-M. Voigt, W. Ebeling, I. Rechenberg and H.-P. Schwefel (eds.), September 22–26, Springer-Verlag, Berlin, Germany, 1996, pp. 42–51.
- 20 C. Jacob, "Evolving evolution programs: Genetic programming and L-systems," in *Proceedings of the First Annual Conference on Genetic Programming*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo (eds.), The MIT Press, Cambridge, Massachusetts, 1996, pp. 107–115.
- 21 C. Jacob, "Evolution and coevolution of developmental programs," *Computer Physics Communications*, pp. 46–50, 1999.
- 22 C. Jacob, *Illustrating Evolutionary Computation with Mathematica*, Morgan Kaufmann Publishers, San Francisco, California, 2001.
- 23 E. D. Jong, "The incremental pareto coevolution archive," in *Proc. of the Genetic and Evolutionary Computation Conference GECCO-2004*, 2004, pp. 525–536.
- 24 J. T. Kim, "Lindevol: Artificial models for natural plant evolution," *Künstliche Intelligenz*, vol. 1, pp. 26–3, 2000.
- 25 G. Kókai, Z. Tóth and R. Ványi, "Application of genetic algorithms with more populations for Lindenmayer systems," in *International ICSC Symposium on Engineering of Intelligent Systems EIS'98*, E. Alpaydin and C. Fyfe (eds.), February 11–13, 1998, University of La Laguna, Tenerife, Spain, Canada/Switzerland, ICSC Academic Press, 1998, pp. 324–331.
- 26 G. Kókai, Z. Tóth and R. Ványi, "Evolving artificial trees described by parametric L-systems," in *Proc. of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering*, Shaw Conference Center, IEEE Press, Edmonton, Alberta, Canada, 1999, pp. 1722–1727.
- 27 J. R. Koza, "Genetic programming," *On the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge, Massachusetts, 1992.
- 28 J. R. Koza, "Discovery of rewrite rules in Lindenmayer systems and state transition rules in cellular automata via genetic programming," in *Symposium on Pattern Formation (SPF-93)*, Claremont, California, 1993.
- 29 J. R. Koza, "Genetic programming II," in *Automatic Discovery of Reusable Programs*, The MIT Press, Cambridge, Massachusetts, 1994.
- 30 S. Luke and R. P. Wiegand, "Guaranteeing coevolutionary objective measures," in *Foundations of Genetic Algorithms VII*, Morgan Kaufman, 2002, pp. 237–251.

- 31 J. McCormack, “Interactive evolution of L-system grammars for computer graphics modelling,” in *Complex Systems: From Biology to Computation*, D. G. Green and T. Bossomaier (eds.), ISO Press, Amsterdam, 1993, pp. 118–130.
- 32 G. F. Miller and D. Cliff, “Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics,” in *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, D. Cliff, P. Husbands, J. Meyer and S.W. Wilson, (eds.), The MIT Press, Cambridge, MA, 1994, pp. 411–420.
- 33 K. J. Mock, “Wildwood: The evolution of L-system plants for virtual environments,” in *International Conference on Evolutionary Computation, Anchorage, Alaska*, 1998, pp. 476–480.
- 34 K. J. Niklas, “Computer-simulated plant evolution,” *Scientific American*, vol. 254, pp. 68–75, 1986.
- 35 S. Nolfi and D. Floreano, “Co-evolving predator and prey robots: Do ‘arms races’ arise in artificial evolution,” *Artificial Life*, vol. 4, pp. 311–335, 1998.
- 36 G. Ochoa, “On genetic algorithms and Lindenmayer systems,” in *Parallel Problem Solving from Nature — PPSN V*, Springer-Verlag, Berlin, 1998, pp. 335–344.
- 37 OpenGL Architecture Review Board, M. Woo, J. Neider, T. Davis and D. Shreiner, *OpenGL Programming Guide: The Official Guide to Learning OpenGL*, Version 1.2. 3rd edition, Addison-Wesley, Reading, Massachusetts, 1999.
- 38 K. Perlin, Noise, “Hypertexture, antialiasing and gesture,” in *Texturing and Modeling: A Procedural Approach*, 2nd Edition, AP Professional, D. S. Ebert, F. Kenton Musgrave, D. Peachey, K. Perlin, and S. Worley (eds.), Cambridge, 1998, pp. 209–274.
- 39 P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer Verlag, New York, 1990.
- 40 T. S. Ray, “An approach to the synthesis of life,” in *Artificial Life II: SFI Studies in the Sciences of Complexity*, C. G. Langton, C. Taylor, J. Doyne Farmer and S. Rasmussen (eds.), Addison-Wesley, Redwood City, CA, vol. X, pp. 371–408, 1991.
- 41 T. S. Ray, “Is it alive or is it GA,” in *Proceedings of the Fourth International Conference on Genetic Algorithms, University of California*, R. K. Belew and L. B. Booker (eds.), Morgan Kaufmann Publishers, San Diego, San Mateo, California, 1991, pp. 527–534.
- 42 T. S. Ray, “Synthetic life: Evolution and optimization of digital organisms,” in *Scientific Excellence in Supercomputing: The 1990 IBM Contest Prize Papers, Athens, Georgia*, K. R. Billingsley, H. U. Brown III and E. Derohanes (eds.), The Baldwin Press, 1992, pp. 489–531.
- 43 T. S. Ray, “An evolutionary approach to synthetic biology: Zen and the art of creating life,” in *Artificial Life: An Overview*, C. G. Langton (ed.), The MIT Press, Cambridge, Massachusetts, 1995, pp. 179–209.
- 44 C. W. Reynolds, “Competition, coevolution and the game of tag,” in *Artificial Life IV*, R. Brooks and P. Maes (eds.), The MIT Press, Cambridge, MA, 1994, pp. 59–69.
- 45 M. Ridley, *The Red Queen: Sex and the Evolution of Human Nature*, Penguin Books, New York, 1994.
- 46 M. L. Rosenzweig, J. S. Brown and T. L. Vincent, “Red queens and ESS: The coevolution of evolutionary rates,” *Evolutionary Ecology*, vol. 1, pp. 59–94, 1987.
- 47 J. Maynard Smith and G. R. Price, “The logic of animal conflict,” *Nature*, vol. 246, pp. 15–18, 1973.
- 48 K. O. Stanley and R. Miikkulainen, “The dominance tournament method of monitoring progress in coevolution,” in *GECCO-2002: Genetic and Evolutionary Computation Conference Workshop Program*, 2002, A. Barry (ed.), pp. 242–248.
- 49 N. C. Stenseth and J. Maynard Smith, “Coevolution in ecosystems: Red queen evolution or stasis,” *Evolution*, vol. 38, pp. 870–880, 1984.
- 50 L. Van Valen, “A new evolutionary law,” *Evolutionary Theory*, vol. 1, pp. 1–30, 1973.