COMPUTER RECREATIONS

Simulated Evolution: wherein bugs learn to hunt bacteria

by A. K. Dewdney

"For those, like me, who are not mathematicians, the computer can be a powerful friend to the imagination."
—RICHARD DAWKINS,
The Blind Watchmaker

On the muddy bottom of a stagnant pool of water a number of protozoa creep about, feeding on the bacteria that slowly rain down on them. The protozoa all look alike, but their behavior shows important differences. Some of them move erratically in search of bacteria and consequently eat little. Others move with more purpose, following a search pattern that seems almost methodical; they find plenty to eat. Such microscopic worlds have a fascination all their own, but this particular scene has special significance: the methodical protozoa evolved from their erratic cohorts in the space of only one hour!

As some readers may already have guessed, such a scene is not viewed through a microscope but on the display screen of a computer. It is generated by a program called Simulated Evolution that was written by Michael Palmiter, a high school teacher from Temple City, Calif. Tiny white protozoan creatures, which Palmiter calls bugs, crawl about on the screen, gobbling up purple bacteria. As generations of bugs pass by, one can watch new feeding behaviors evolve.

Richard Dawkins of the University of Oxford has also looked for insights into evolution by investigating programs that attempt to simulate its various aspects. One such program, written by Dawkins himself, was this department's subject more than a year ago [see SCIENTIFIC AMERICAN, February, 1988]. Dawkins' program displays biomorphs: computer-generated forms that sometimes resemble living creatures. They evolve by a process of "artificial selection": the computer operator arbitrarily selects one of nine possible variant forms of the current biomorph as the basis for future generations of biomorphs.

The biomorphs that emerge from Dawkins' program can be bizarre and amusing—and sometimes even lifelike—but they cannot be said to have evolved naturally, that is, under internal selective pressures. Yet Dawkins thinks it ought to be possible to write a computer program that mimics natural selection. Computer-generated species having such "evolvability" would radiate increasingly complex forms, which selection would pare to a manageable number. Moreover, the surviving descendants would then have to be capable of evolving in new ways that were completely unavailable to ancestors.

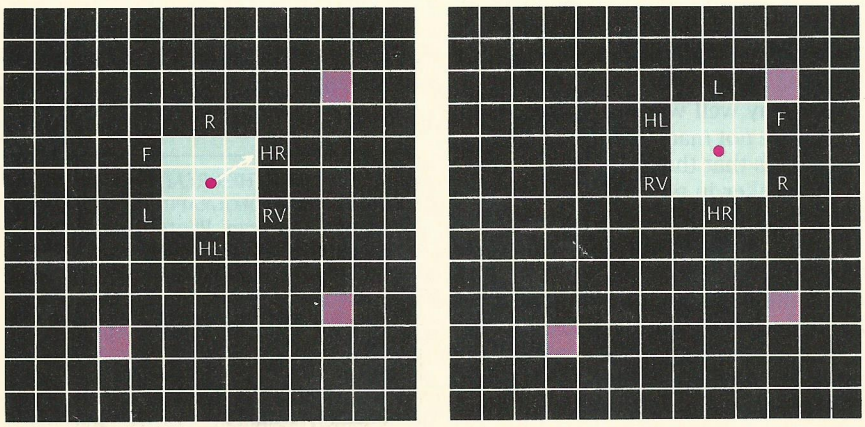
Palmiter's protozoan bugs certainly move us one step closer to Dawkins' goal. As a glance at the illustration on the opposite page reveals, the bugs (the white blips) live within a rectangle on which bacteria (the purple blips) are continually being deposited. The bugs pursue a life dominated by moving and feeding on bacteria. Each bacterium eaten by a bug provides the bug with 40 units of energy, which is enough to make 40 moves. In places where the feeding is rich, a bug may readily acquire 1,500 units of energy in a few minutes. If that happens, however, a strange mechanism kicks in: all further eating does not benefit the bug until its energy level falls below 1,500 units.

On the other hand, it may happen that a bug finds very little to eat over an extended period. In such a case the bug's energy reserves could gradually drop to zero. At that point it would appear to sit morosely for a few cycles, as though pondering its end, and then wink out like a small light.

A bug's success in finding food depends, of course, on the relative abundance of bacteria in its immediate neighborhood. Because the bacteria are deposited more or less uniformly within the white rectangle, if localized feeding has depleted the bacteria in one place, they are bound to be plentiful elsewhere. Some bugs appear to get to the areas of relative abundance quicker than others. It all depends on the moves a bug makes—its search pattern, so to speak.

The Darwinian scenario of the program Simulated Evolution, albeit abstract, hinges on the "genes" that govern the way a bug moves. These particular genes probably do not exist in real protozoa, but Palmiter's bugs have six of them. They are labeled *F, R, HR, RV, HL* and *L* for Forward, Right, Hard Right, Reverse, Hard Left and Left. (All directions are expressed from the bug's point of view. Normal turns amount to 60 degrees in one or the other direction, whereas hard turns are 120 degrees.)

On any given move the bug heads in a direction chosen by lottery: the program picks one of the six possible directions from a kind of mathemati-



A bug's turns are relative to its current direction

cal hat. If the program chooses *L*, for example, the bug makes a 60-degree turn to the left. The probability that a particular direction will be chosen is given by a value assigned to the corresponding gene. Hence the higher a gene's value, the greater its contribution to the bug's overall pattern of movement. If, for example, a bug has a large *L* value in relation to the other five gene values, the bug will spend a lot of time veering to the left.

Every possible combination of gene values results in a different general pattern of movement, and whatever a bug's genetic makeup may be, the bug is stuck with it for life. It can only hope (to be somewhat anthropomorphic) that its offspring will do better.

After a bug has made 800 moves, it becomes "mature" and is ready to reproduce. It does so only if it also happens to be "strong," that is, if it has 1,000 or more units of energy stored under its electric-white membrane. Paramecia undergo a process called conjugation when they reproduce, but the bugs fission: a strong mature bug splits into two new ones, each with half the energy of its parent. When that happens, the new bugs inherit the movement genes of the parent but with a small difference. The value of one of the genes in each offspring is increased or decreased slightly.

Suppose, for example, a strong mature bug has the gene values $F=3$, $R=2$, $HR=0$, $RV=-2$, $HL=0$ and $L=1$. Its two offspring, labeled *A* and *B*, might inherit the following, mutated forms of that genetic makeup:

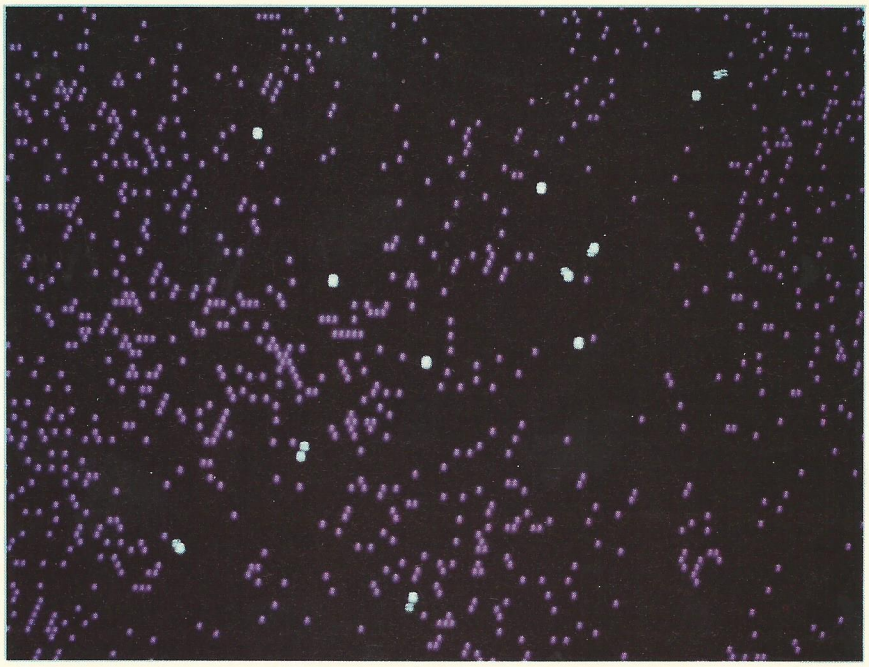
A: $F=4$ $R=2$ $HR=0$ $RV=-2$ $HL=0$ $L=1$

B: $F=3$ $R=2$ $HR=0$ $RV=-2$ $HL=-1$ $L=1$

As can be seen, in offspring *A* the *F* value has been incremented by 1, and in offspring *B* the *HL* value has been decremented by 1.

How will the offspring differ from their parent? Offspring *A* will have a slightly greater tendency to move forward than its parent did, whereas offspring *B* will have a slightly lesser tendency to make hard-left turns. Such small shifts in tendencies are barely perceptible on the computer screen to a trained observer.

In its simplest mode, Simulated Evolution starts out by endowing 10 bugs with a random genetic structure, which causes the great majority of them to jitter from side to side in an unpredictable manner. As a rule such "jitterbugs" exhibit a high death rate. They simpy tend to eat up most of the food in their immediate vicinity and then jiggle themselves into starvation on barren ground. Nevertheless, some do survive.



"Jitterbugs" slowly evolve into "cruisers"

Generation succeeds generation every minute or so. This miniature life-and-death struggle makes for absorbing viewing, but the drama is greatly heightened after several minutes, when the viewer becomes aware that some of the bugs have begun to behave differently. They do not jitter; they bobble. Then, a few minutes later, there are bugs that tumble. After 20 minutes or more one can see bugs that glide—at least for short distances. These bugs appear to do much better than their jittery ancestors. Indeed, they proliferate before one's very eyes for precisely that reason.

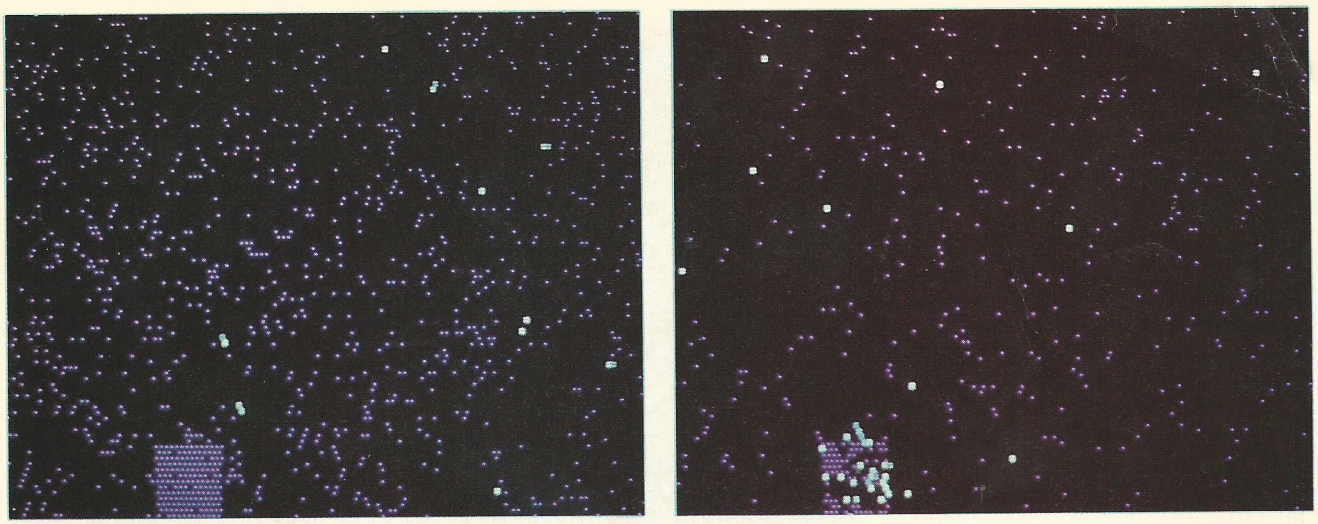
In due course "cruiser" bugs develop that move forward most of the time but turn every now and then. This means they are almost always moving toward denser populations of delicious purple bacteria. Once the behavior is established in just a few individual bugs, it comes to dominate the entire population, since the cruisers end up gathering the lion's share of the food.

Although the cruisers constitute a species of sorts, there nonetheless is still some variation within the cruiser population. For example, some cruisers turn more often to the right than to the left whereas others favor left turns. There are also occasional setbacks, of course. Some cruisers spawn maladapted descendants. A common genetically transmitted disease is the "twirlies," wherein a bug makes too many turns in one direction. Such unfortunate creatures usually die without having known the joy of fission.

It is interesting enough to watch the cruiser species emerge, but Palmiter's program offers more. What if there is variation in the environment? Will more than one species evolve? That question is answered by running Simulated Evolution in a mode in which the screen looks much the same except for a particularly rich patch of bacteria in the lower left-hand corner. The bacteria in that patch are replenished at a much higher rate than normal [*see illustration on next page*]. Palmiter calls that bountiful area the Garden of Eden.

As generations of bugs come and go, the cruisers evolve as before. But within the Garden of Eden something quite different happens. A few lucky jitterbugs that have stumbled into the bacterial banquet are promptly rewarded for their lack of an organized feeding method. Jiggle as they will, food continues to surround them.

As food becomes scarcer in the Garden of Eden, however, a subtle environmental pressure begins to operate. Jiggling and jittering soon are no longer viable strategies. That is when the twirlers make their appearance. What normally is a disastrous genetic defect is actually an advantage in an overpopulated Garden of Eden. Indeed, in the course of time those bugs with a strong tendency to turn in one direc-



***The Garden of Eden* (left) *fosters the evolution of "twirlers"* (right)**

tion predominate in the garden. The reason is obvious. A bug that turns frequently in the same direction, say to the right, will tend to remain in the Garden of Eden longer than its jittery ancestors.

Within a few hours at most the Garden of Eden is populated almost exclusively by highly specialized twirlers that might as well be called nervous orbiters. They follow a specific orbit for many cycles and then suddenly move just one square away and repeat the orbit, sweeping up bacteria with each shift.

Is the Simulated Evolution program a valid model of biological evolution? Only in a very limited sense. It shows how an environment can favor certain variations in offspring, leading ultimately to the formation of new species. But that is as far as the similarities go. Once one or two stable bug species have emerged, nothing else happens. What would it take to realize Dawkins' dream of an indefinitely continuing computer-generated evolution? Perhaps nothing less than a miniature universe inside the computer.

Readers who would like to study the subject can order a copy of Simulated Evolution for $39.95 from Life Science Associates, a small educational-software company. The address is 1 Fenimore Road, Bayport, N.Y. 11705. The program runs on IBM PC and compatible computers, and it comes with an elaborate manual. For those relatively advanced programmers who prefer to write their own version of Simulated Evolution, I shall now describe BUGS, my name for a simplified version of the program.

A BUGS bug can be represented by a small square that has three pixels on a side. The six directions in which such a bug moves can then be illustrated as they are on page 138. A simple table specifies how the coordinates of a bug's central pixel change depending on the direction in which the bug is heading. The table contains two six-element arrays, *xmove* and *ymove*.

dir	0	1	2	3	4	5
xmove	0	2	2	0	-2	-2
ymove	2	1	-1	-2	-1	1

The direction in which a bug is heading (with respect to the computer screen) is given by the value of the variable *dir*. The corresponding numbers in *xmove* and *ymove* indicate by how many pixels, in the horizontal and vertical directions respectively, the bug must accordingly be shifted on the screen in a single move. If, for example, a bug is headed in direction 2, it must be shifted to the right by two pixels and down by one pixel, since $xmove(2) = 2$ and $ymove(2) = -1$.

BUGS determines the direction of motion for each of the creatures in its charge by consulting a formula based on each bug's genetic code, which is contained in a two-dimensional array called *gene*. The element $gene(k,j)$ contains the jth gene value of the kth bug. In the formula each gene value is made an exponent of 2 in order to avoid having to deal with negative numbers. The probability that a bug will move in direction d is then found by dividing 2 raised to the d-gene value by the sum of 2 raised to each of the gene values. For example, the probability that the bug will next turn hard left is the result of dividing 2^{HL} by the sum $2^F + 2^R + 2^{HR} + 2^{RV} + 2^{HL} + 2^L$.

In this manner BUGS calculates six probabilities for each of the possible moves. When the probabilities are added up, the result naturally is 1. One can think of the probabilities as six different ranges that together span a number line extending from 0 to 1. In other words, if the probabilities for the six different directions of motion are represented by p_0 through p_5, then range 0 consists of the interval from 0 to p_0, range 1 consists of the interval from p_0 to $p_0 + p_1$, range 2 consists of the interval from $p_0 + p_1$ to $p_0 + p_1 + p_2$ and so on.

In each cycle BUGS then determines a new value for the direction of motion for a particular bug by selecting a random number between 0 and 1, seeing in which range it has happened to fall and assigning the range number to the variable *turn*. In this scheme, then, *turn* equals 0 for *F*, 1 for *R*, 2 for *HR*, 3 for *RV*, 4 for *HL* and 5 for *L*.

A few simple statements complete the motion algorithm:

$$dir \leftarrow dir + turn \pmod 6$$
$$bugx(k) \leftarrow bugx(k) + xmove(dir)$$
$$bugy(k) \leftarrow bugy(k) + ymove(dir)$$

The arrays $bugx(k)$ and $bugy(k)$ contain the kth bug's current coordinates.

In the first line the current direction *dir* is changed by adding the result of the turning lottery embodied in the variable *turn*. Addition must be modular. For example, if $dir = 5$ (which means the bug is heading up and to the left) and $turn = 2$ (which means it needs to turn hard right), the new value for *dir* will be $5 + 2 \pmod 6 = 1$, and the bug's next move is up and to the right.

BUGS must move all bugs according to this formula, at each step checking whether a bug has hit a barrier or landed on a bacterium. In addition it

must keep a record of each bug's age and energy supply in order to determine whether a particular bug should be extinguished or allowed to fission. When a bug is ready to fission, the program merely replaces the old bug with two new ones at the same location. These inherit the old bug's gene values except that a randomly selected gene value is increased by a certain amount in one offspring and another randomly selected gene value is decreased by the same amount in the other.

This description of BUGS will be enough for some to try their hand at writing the program. Those who find the description a bit spare may order a more detailed algorithmic outline from me, enclosing a check or money order for $2 to cover costs.

Enthusiasm for the fractal-generating program SLO GRO, which I described in last December's issue, did not grow slowly. A hefty bag of mail hinted at the continuing interest in fractals in any shape or form. The program can be described scientifically as a simulator of the diffusion-limited aggregation, something we see in the formation of certain minerals, electrolytic plating of metals and even in the accumulation of soot.

The SLO GRO recipe was sufficiently simple for many readers to follow, and many in fact did so. The basic algorithm involves the injection of a randomly walking "particle" into a circle from a random point on the circle's circumference. When the particle comes in contact with a stationary fellow particle, it too ceases to move and thus produces an aggregation of particles. The program was easy to write but was somewhat painful for certain people to watch. Why should they spend their time watching a point of light jittering for what seemed forever? As a result several readers thought of changes in the algorithm that speeded its operation.

Edward H. Kidera IV of Columbia, Md., achieved a definite speedup by starting with a small circle and steadily increasing its radius as the aggregation grew. A number of readers also made suggestions for speeding up the test for contact with a crowd of fellow particles. The test involved comparing the particle's neighboring pixels with the recorded positions of every particle in the growing crowd.

Ronald C. Read of the University of Waterloo in Ontario made the following suggestion on this very point. "For those who use BASIC (as I'm sure many of your readers do) there is a much easier way. That is to use the POINT command of BASIC in order to tell whether the pixel in question has been given a color. In effect, then, one is using the screen as a storage device."

Most impatient of all was William H. Pratt of State College, Pa. Why make the particle wander randomly at all? Why not just give it a random position next to the growth itself? Pratt was dismayed, however, to find that his growth looked nothing like last December's illustrations. It certainly was ragged about the edges but more solid—a different creature altogether.

Pratt was unknowingly playing with what is known as Richardson's growth model, a favorite research tool of a group of mathematicians called "the particle mafia." These investigators, some of whom are based at the University of Wisconsin at Madison, have been studying a great variety of growth models for more than a decade. I hope to report on a recent visit to Madison in a future column.

In January this department featured people puzzles: logic puzzles that can be solved only by thinking about what other people are thinking. An entire class of such puzzles was represented by three philosophers who awoke from an afternoon slumber under a tree. Each philosopher noted that the foreheads of the other two had apparently been befouled by a bird. Only in the course of the ensuing laughter did the wisest of them realize that his own forehead was decorated. How did he make the deduction?

It had not occurred to me, as it did to James D. Klein of College Place, Wash., that there is a two-philosophers puzzle of sorts. Klein tested his own children with the story of a pair of workmen who fall from a scaffold onto the ground. The fall does not hurt either of them, but it does dirty the face of one. Why did the workman with the clean face rush to wash up while the one with the dirty face merely went back to work? Klein writes, "It is interesting to hear them think out loud and watch their eyes as the solution dawns."

Another people puzzle was borrowed from Dennis Shasha's book *The Puzzling Adventures of Dr. Ecco.* In this conundrum two 19th-century generals whose armies are separated by a ridge of land decide to coordinate their attack on the enemy by sending messages by carrier pigeon. But what message to use? If the first general sends the message "Attack at dawn," he must wait for a return message from the second general to confirm that he has received it. What if one of the pigeons never makes it to the other side? And even if both pigeons arrive at their destinations, how does the second general know that his confirmation has been received? An infinite regress of messages appears to be inescapable.

The generals' predicament reminded Warner Clements of Beverly Hills, Calif., of a little-known off-Broadway play that involved a would-be double agent shuttling back and forth between two hostile nations. It begins when the agent learns that country *A* has broken the secret military code of country *B.* The agent goes to *B* in order to sell that country's intelligence officers the information. "We already know that," the officers say. The agent is at first discouraged but then realizes he can sell that information to the intelligence officers of country *A.* They in turn reply, "We know the *B*'s have broken our code. We have been sending them false information!" The agent rushes back to country *B:* "Do you realize the *A*'s know you have broken their code?" "Oh yes," reply the *B* officers. The agent returns to the *A*'s to apprise them of the situation, and so on. How long might the agent have to continue the back-and-forth journeys, bearing an ever lengthening message about what the other side knows? Although in this puzzle the two military factions are not coordinating but competing, it makes the solution no easier—there not being one.

More down-to-earth people puzzles involved real people in everyday situations like those studied by the late Erving Goffman, a sociologist. I asked for examples and received several, including one from P. M. Cambeen of Muiden in the Netherlands. During World War II an officer in the German force occupying Holland expressed to a resident his puzzlement at Dutch people's attitudes. He was told: "The Dutch have three virtues. They are intelligent, loyal and pro-Nazi. Any given Dutch person, however, has only two of these virtues and the opposite of the third." While the logical implications of his statement were being worked out by the officer, the wit had enough time to get away.

FURTHER READING

THE BLIND WATCHMAKER. Richard Dawkins. W. W. Norton & Company, 1987.

ARTIFICIAL LIFE: THE PROCEEDINGS OF AN INTERDISCIPLINARY WORKSHOP ON THE SYNTHESIS AND SIMULATION OF LIVING SYSTEMS HELD SEPTEMBER, 1987, IN LOS ALAMOS, NEW MEXICO. Edited by Christopher G. Langton. Addison-Wesley Publishing Company, Inc., 1989.