Chapter 9

Hominids

Of the animals homo sapiens sapiens (us) have the largest brain/per body weight and the most complex behaviors. It is puzzling as to just what the big brain was for as measures of skill such as tool use show flat development during the millions of years that the brain was increasing in size. This may have happened through sexual selection, which produced a runaway growth in brain size. Genetic algorithms show that runaway affects are the rule.

How did humans get large brains? This is a fundamental question that is still in the process of being answered. We visit it here because computational models are increasingly playing a role in ferreting out the answer.

Figure 9.1 shows the brain weights of humans and their ancestors compared to those of living animals. The mammals are characterized by the purple lines which are substantially above the frogs and reptiles. This is because the mammals have corticies and the attendant forebrain structures. Not shown on this graph but near the top with highly developed brains are the former land mammals, dolphins and whales. Stunningly recent work suggests that dolphins have individual names that they use to identify themselves in calls. So far they are the only animals besides ourselves that have been shown to do this.

9.1 The fossil record on hominids

How old are humans? This question can be answered by studying *mirochondrial DNA*. Every cell contains chromosomes that consist of large strands of

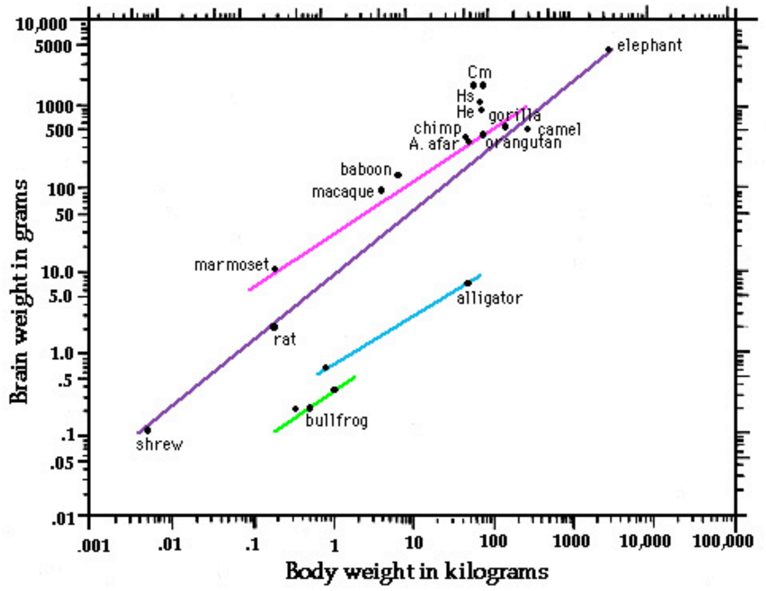

Figure 9.1: Comparison of brain weight versus body weight for human ancestors compared to living animals. *Cm* - common man. *Ho* - homo erectus. *Hs* - homo sapiens. *A afar* - Australopithecus afarensis [Permission Pending]

the DNA molecule that contains the material for copying the cell. But it also contains mitochondrial DNA. During reproduction the DNA in the nucleus is divided up, some coming from the male parent and some from the female parent. However mitochondrial DNA is special in that it is copied *in toto* from the female parent. Parenthetically, you might wonder why this is idiosyncratically so. It is believed that this DNA is a result of an invasion into a prehistoric single cell by bacteria. The invasion was unsuccessful, the bacteria got stuck and was used for extra energy by the cell. A speculation is that this extra energy may have allowed cells to specialize into larger cell colonies an ultimately us. But the main point here is that, since this DNA is copied without alteration, the only alterations present can come from mutations introduced by cosmic radiation. Thus the notion of a 'mitochondrial Eve.' [1] The mitochondrial DNA that we all share only differs by mutations, so that if one traces back far enough, one has the concept of a common mother from which we are all descended. The trick is estimating the mutation rates, but one estimate puts this date for the mitochondrial eve at 200,000 years ago, making homo sapiens sapiens very recent on the evolutionary time scale.

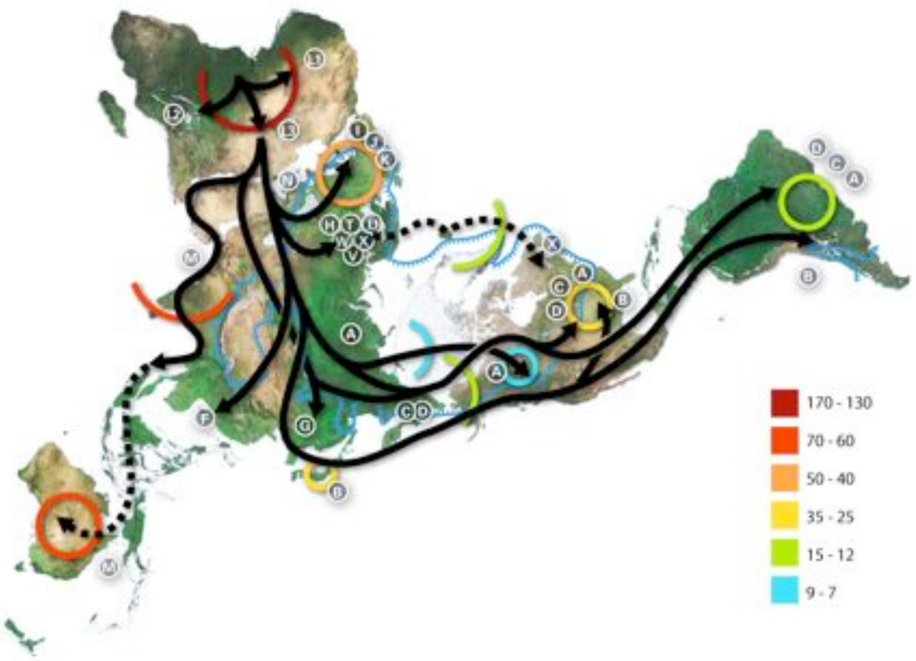

Figure 9.2: Migration of early humans based on mitochondrial DNA data. Letters denote distinct groups. [Permission Pending]

Of course a lot happened in these 200,000 years. Elaborate camps with primitive but specialized bone tools date from about 70,000 years ago. And the every detailed and exquisite cave paintings in France date from 40,000 years ago. The huge trappings of civilization date from about 5,000 to 10,000 years ago with the invention of writing and formation of large cities.

By the standard of these developments those of the previous years seem modest. Figure 9.3 shows brain size plotted for the time period spanning three million years before the present (YBP). At three million YBP human ancestors live in Africa and are bipedal. At two million YBP they are making very primitive stone axes that are thought to have been used for scraping hides and meat from the carcasses of animals. A prevailing set of ideas is that climate change forced them from arboreal environments onto grassy savannas where they had to compete with large carnivores for food. They way they could do that is to scavenge meat from carcasses as hyenas do today. In addition living in groups became more important for safety.

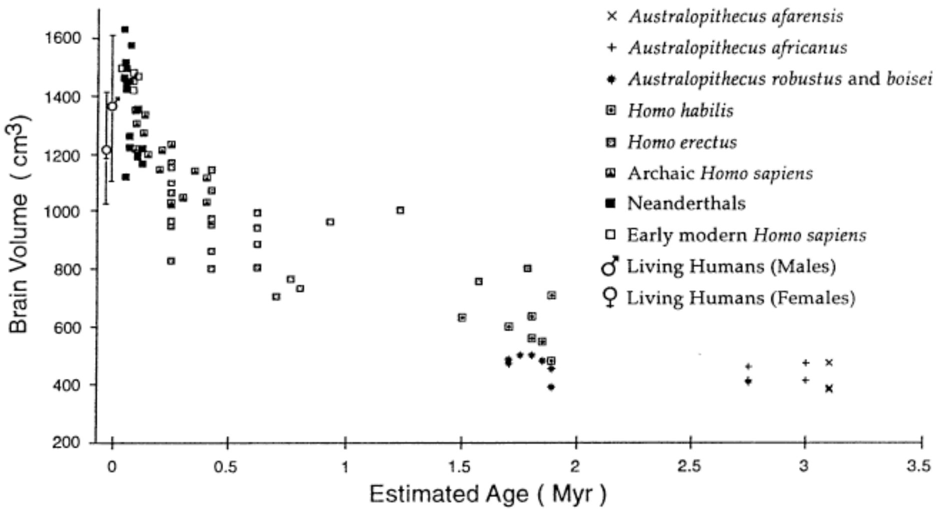

FIG. 4. Brain size (in cm^3) plotted against time (Myr) for specimens attributed to Hominidae.

Figure 9.3: Brain size represented in terms of volume plotted against age. In the figure time, represented as millions of years before the present, is plotted from right to left. The different symbols represent data from different human species. [Permission Pending]

Al this time brain size is increasing steadily but does not really take off to its present level of 1400 cubic centimeters until about 500,000 YBP.

9.2 DNA as program instructions

At the time of Mendel the understanding of genes was much simpler than now. The packaged genetic material could be thought of as consisting of genes that were expressed in traits such as eye color. Eye person has two copies of each gene, the copies being termed *alleles.* During reproduction, one allele from each parent appears in the new individual and only one of these is expressed in the phenotype. So the chance of getting any particular gene from a parent is one forth. You have half a chance that it will be copied and half a chance that it will be expressed. Figure 9.4 shows this possibility.

This description still works but the situation is much more complicated. The process of making a phenotype is governed by proteins. Estimates are

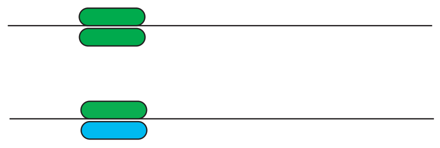

Figure 9.4: In a highly schematized version of genes, they appear in a segment of DNA in pairs. During reproduction just one of these is copied and just one of the copied genes is expressed.

that the of the number of proteins is about 30,000 or so. Different proteins are made by 'reading' the genetic code and constructing proteins as strings of twenty simpler amino acids. You could think of a protein as an amino acid 'necklace' and that would work up to a point. The essential missing feature in that description is that the proteins work by each having very complicated three-dimensional structure that they manipulate to direct chemical reactions. So the bad news is that the situation has become much more complicated but the good news is that we are beginning to understand parts of it.

The other astonishing bit of news is that the DNA is not a static piece of data that is read in the making of proteins but contains chemical switches. To understand this part, it helps to appreciate the 'before picture.' At one time, until fairly recently, an old saw was that the brain had to rely heavily on the environment - nurture- for its programming as their were not even remotely enough bits in the DNA to encode the brain's structure. A standard argument was that the number of bits in the DNA was much less that the number of bits needed to specify synapse weights. Now we know that the situation is different because the proteins can interact with the DNA in the course of developing a phenotype by turning on and off switches on the DNA. This capability allows the progression of protein manufacture to be modulated in time. This is easily appreciated by considering the concept of stem cells. Since phenotypes start from a single cell, initially all the cells look alike, but as they divide, their individual chemical switches get set in different ways so that they can develop into different cells with very different properties. In short the entire process can be thought of a a very large computer program!

The fact that the DNA represents a piece of program code is illustrated by an astonishing manipulation done by Gehring and colleagues. They were able to transplant a piece of the DNA that specifies a mouse eye into a segment of the DNA of a small fly known as *drosophila.* To everyone's amazement compound eyes appeared on the drosophila. These eyes were non-functional and appeared at strange places, but they were compound eye structures as shown in Figure 9.5. This experiment overturned another popular belief and that was that since vision was so important it had been evolved separately in insects and mammals. Insect eyes have a tessellation individual lenses where each lens has its own set of neurons at the end. In contrast, mammals have a single lens that focuses light on an array of light sensitive cells. The argument was that the eyes were so different in structure, that they would have very different DNA specifications. But the experiment shows just the opposite, the program code that works for the mouse works for the fly also.

9.3 Genetic Algorithms

While it will take a long time before we understand the complicated program that constructs the phenotype, the forces that guide evolution have many insightful models that are cast as computer programs. The general class of algorithms that are based on evolutionary ideas are naturally termed *genetic algorithms.* Such algorithms describe their DNA equivalent as a string of symbols and the fitness function as a function that produces a number for each symbol instance. A more complicated variant is *genetic programming* or GP. In GP, the symbol string represents a program that can be executed. The performance of the program can be rated and that rating becomes its fitness.

The basic idea of a genetic algorithm is very simple. Instances of the genotype are coded as strings. The algorithm uses a population of these strings where different strings have different fitness values. In the actual biology fitness is an incredibly complex function that represents the planet's ecosystem including all the competitive species with which the population shares territory. However here all that complexity is encoded into a single, usually fairly simple, function that can produce a fitness number for each string.

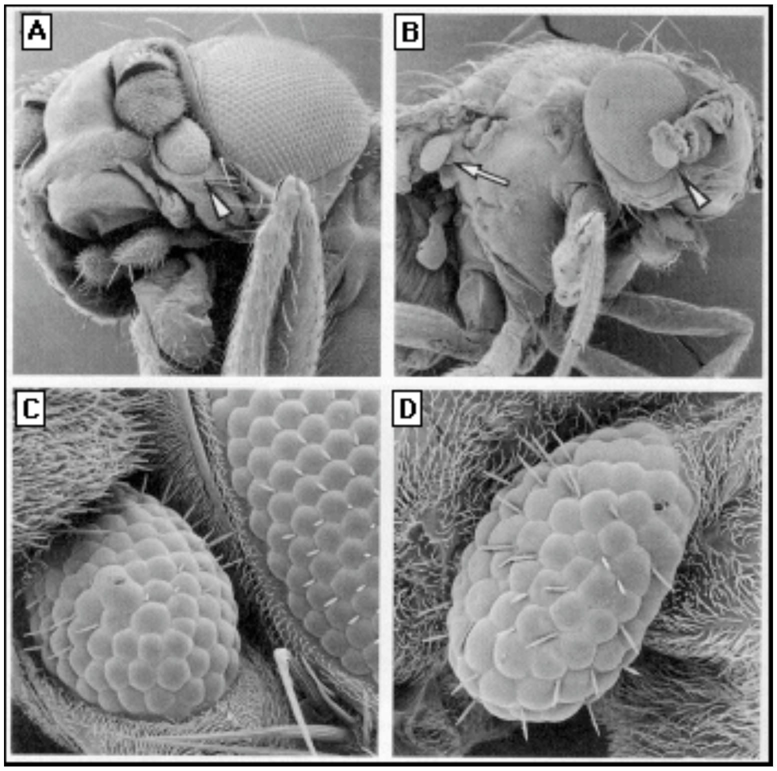

Figure 9.5: Electron micrographs of drosophila showing the compound eyes that result from a transplanted mouse gene. A and B show large scale views and C and D show close ups.

The simplest versions of GAs do not worry about having males and females. Instead pairs of individuals are selected randomly based on their fitness. Higher fitness individuals are more likely to be selected. New individuals are produced by a crossover operation that interchanges segments of the strings. Usually the algorithm is divided into generations where a number of new individuals are produced, a number of individuals are selected for death, and then the resultant population has their fitness values recomputed and the process repeats. To make this all clear we will walk through a particular example.

The example is purposely chosen to be very simple so all the basic steps can be illustrated. Consider finding the maximum of the function

$$f(x) = -x^2 + 12x + 300$$

Table 9.1: Initial condition for the genetic algorithm example.

Individual	**Genetic Code**	x	$f(x)$	$p(x)$
1	10110	22	80	0.08
2	10001	17	215	0.22
3	11000	24	12	0.01
4	00010	2	320	0.33
5	00111	7	335	0.36
Average			192	

where x takes on integer values $0, \ldots, 31$. This example is easily solved with direct methods, as any college calculus graduate will tell you, but it will be encoded as a GA search problem to demonstrate the operations involved.

The first step in defining the GA is to code the search space. The encoding used for genetic operations is the binary code for the independent variable x. The numbers 0 to 31 need five binary bits where 00000 is zero, 00001 is one, 00010 is two, 00011 is three and so on. The binary code is a very simple representation of the 'DNA' for this problem.

The actual number that each string represents can be thought of as a simple 'phenotype.' Thus the fitness of each number is just that returned by the fitness function $f(x)$. An auxiliary step is needed to turn the fitness value into a probability, but this is easily accomplished by normalizing over the individuals. Thus the probability of being selected for mating $p(x)$ is specified by

$$p(x) = \frac{f(S)}{\sum_i f_i}$$

The normalized fitness over the whole population determines the probability of being selected for reproduction. Table 9.1 shows an initial condition for the algorithm starting with five individuals.

Now select two individuals for reproduction. Selections are made by accessing a random number generator, using the probabilities shown in the column P_{select}. Suppose that individuals 2 and 4 are picked.

The operation we will use is crossover, which requires picking a locus on the string for the crossover site. This is also done using a random number generator. Suppose the result is two, counting the front as zero. The two new individuals that result are shown in Table 9.2 along with their fitness values.

Table 9.2: Mating process.

Mating Pair	**Site**	**New Individual**	$f(x)$	$p(x)$
00010	2	10010	192	0.14
10001	2	00001	311	0.23

Table 9.3: The genetic algorithm example after one step.

Individual	**Genetic Code**	x	$f(x)$	$p(x)$
1	10010	18	192	0.14
2	10001	17	215	0.16
3	00001	1	311	0.23
4	00010	2	320	0.23
5	00111	7	335	0.24
Average			275	

Now these new individuals have to be added to the population, maintaining population size. Thus it is necessary to select individuals for removal. Once again the random number generator is consulted. Suppose that individuals 1 and 3 lose this contest. The result of one iteration of the GA is shown in Table 9.3.

Note that after this operation the average fitness of the population has increased. You might be wondering why the best individuals are not selected at the expense of the worst. Why go to the trouble of using the fitness values? The reason can be appreciated by considering what would happen if it turned out that all the best individuals had their last bit set to 1. There would be no way to fix this situation by crossover. The solution has a 0 in the last bit position, and there would be no way to generate such an individual. In this case the small population would get stuck at a local minimum. Now you see why the low-fitness individuals are kept around: they are a source of diversity with which to cover the solution space.

9.4 Schemata

In very simple form, the example exhibits an essential property of the genetic coding: that individual loci in the code can confer fitness on the individual. Since the optimum is 6, all individuals with their leading bit set to 0 will be fit, regardless of the rest of the bits. In general, the extent to which loci can confer independent fitness will simplify the search. If the bits were completely independent, they could be tested individually and the problem would be very simple, so the difficulty of the problem is related to the extent to which the bits interact in the fitness calculation.

A way of getting a handle on the impact of sets of loci is the concept of *schemata* (singular: schema). The exposition here follows David E. Goldberg's text *Genetic Algorithms in Search, Optimization, and Machine Learning* (Reading, MA: Addison-Wesley, 1989), which has much additional detail. A schema denotes a subset of strings that have identical values at certain loci. The form of a schema is a template in which the common bits are indicated explicitly and a "don't care" symbol ($*$) is used to indicate the irrelevant part of the string (from the standpoint of the schema). For example, $1*101$ denotes the strings $\{10101, 11101\}$. Schemata contain information about the diversity of the population. For example, a population of n individuals using a binary genetic code of length l contains somewhere between 2^l and $n2^l$ schemata.

Not all schemata are created equal, because of the genetic operations, which tend to break up some schemata more than others. For example, $1****1$ is more vulnerable to crossover than $*11***$. In general, short schemata will be the most robust.

To see the importance of schemata, let's track the number of representatives of a given schema in a population. It turns out that the growth of a particular schema in a population is very easy to determine. Let t be a variable that denotes a particular generation, and let $m(S,t)$ be the number of schema exemplars in a population at generation t. To simplify matters, ignore the effects of crossover in breaking up schema. Then the number of this schema in the new population is directly proportional to the chance of an individual being picked that has the schema. Considering the entire population, this is

$$m(S,t+1) = m(S,t)n\frac{f(S)}{\sum_i f_i}$$

because an individual is picked with probability $\frac{f(S)}{\sum_i f_i}$ and there are n picks. This equation can be written more succinctly as

$$m(S, t+1) = m(S,t)\frac{f(S)}{f_{ave}}$$

To see the effects of this equation more vividly, adopt the further simplifying assumption that $f(S)$ remains above the average fitness by a constant amount. That is, for some c, write

$$f(S) = f_{ave}(1+c)$$

Then it is easy to show that

$$m(S,t) = m(S,0)(1+c)^t$$

In other words, for a fitness that is slightly above the mean, the number of schema instances will grow exponentially, whereas if the fitness is slightly below the average (c negative), the schema will decrease exponentially. This equation is just an approximation because it ignores things like new schema that can be created with the operators, but nonetheless it captures the main dynamics of schema growth. Figure 9.6 shows the main effects clearly. At three different points in the evolution simulation something new is discovered that increases fitness and it quickly spreads through the population.

Of course whether this happens in a real system depends on the resolution of a lot of complications. As Ridley has pointed out [**?**], for an animal the fitness function is often critically dependent on the other species in its habitat. If a predator takes advantage of a particular feature, the prey can evolve a defense to this exploitation, starting a never ending process. Nonetheless if ideal conditions prevail and there are no such corrections, then the exponential take-off shown in Figure 9.6 will occur.

The upshot of the previous analysis has been to show that fit schemata propagate exponentially at a rate that is proportional to their relative fitness. Why is this a good thing to do? The answer can be developed in terms of a related problem, the two-armed bandit problem (Las Vegas slot machines are nicknamed "one-armed bandits"). The two-armed slot machine is constructed so that the arms have different payoffs. The problem for the gambler is to pick the arm with the higher payoff. If the arms had the same payoff on each pull the problem would be easy. Just pull each lever once and

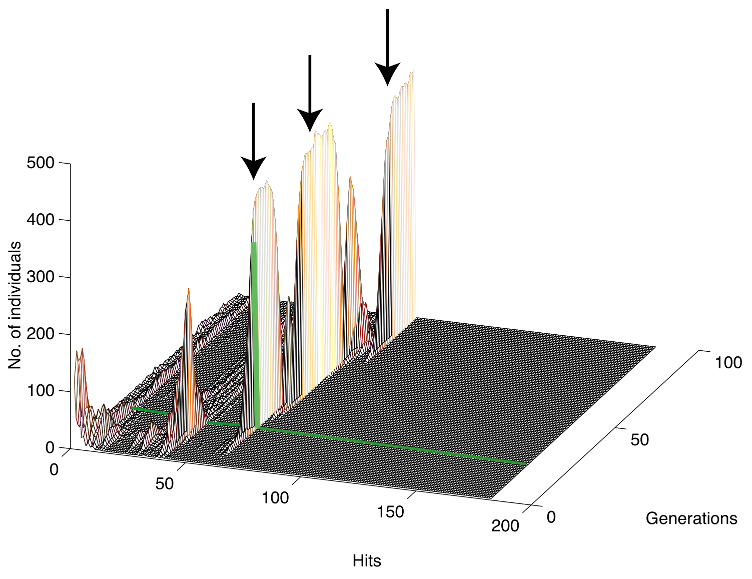

Figure 9.6: An example from a particular genetic program showing the run-away effect when a schema is discovered. In this plot one axis shows the fitness, indicated as 'hits,' and another shows the generations. Thus the plot shows how the fitness distribution among individuals - or fitness histogram - evolves with the generations. For example the green line shows that at generation 20 most of the population has a fitness level near 65. But the important point is that at the indicated arrows something new has been discovered. The time course shows that it quickly spreads throughout the population.

then pull the winner after that. The problem is that the arms pay a random amount, say with means m_1 and m_2 and corresponding variances σ_1 and σ_2.

This problem can be analyzed by choosing a fixed strategy for N trials. One such strategy is to pull both levers n times (where $2n < N$) and then pull the best for the remaining number of trials. The expected loss for this strategy is

$$L(N, n) = |m_1 - m_2|\{(N - n)p(n) + n[1 - p(n)]\} \tag{9.1}$$

where $p(n)$ is the probability that the arm that is actually worst looks the best after n trials. We can approximate $p(n)$ by

$$p(n) \approx \frac{e^{-x^2/2}}{\sqrt{2\pi}x}$$

where

$$x = \frac{m_1 - m_2}{\sqrt{\sigma_1^2 + \sigma_2^2}}\sqrt{n}$$

With quite a bit of work this equation can be differentiated with respect to n to find the optimal experiment size n^*. The net result is that the total number of trials grows at a rate that is greater than an exponential function of n^*. More refined analyses can be done, but they only confirm the basic result: than an exponential function of n^*. More refined analyses can be done, but they only confirm the basic result: Once you think you know the best lever, you should pull that lever an exponentially greater number of times. You only keep pulling the bad lever on the remote chance that you are wrong. This result generalizes to the k-armed bandit problem. Resources should be allocated among the k arms so that the best arms receive an exponentially increasing number of trials, in proportion to their estimated advantage.

In the light of this result let us return to the analysis of schemata. In particular, consider schema that compete with each other. For example, the following schemata all compete with each other:

$$\begin{array}{c}
**0*00* \\
**0*01* \\
**0*10* \\
**0*11* \\
**1*00* \\
**1*01* \\
**1*10* \\
**1*11*
\end{array}$$

Do you see the relationship between the bandit problem and schema? If these schemata act independently to confer fitness on the individuals that contain them, then the number of each schema should be increased exponentially according to its relative fitness. But this is what the GA is doing!

You should recognize that all of the foregoing discussion has not been a proof that GAs work, but merely an argument. The summary of the argument is as follows:

- GAs seem to work. In practice, they find solutions much faster (with higher probability) than would be expected from random search.

- If all the bits in the GA encoding were independent, it would be a simple matter to optimize over each one of the bits independently. This is not the case, but the belief is that for many problems, one can optimize over subsets of bits. In GAs these are schemata.

- Short schemata have a high probability of surviving the genetic operations.

- Focusing on short schemata that compete shows that, over the short run, the fittest are increasing at an exponential rate.

- This has been shown to be the right thing to do for the bandit problem, which optimizes reward for competing alternatives with probabilistic payoffs that have stable statistics.

- Ergo, if all of the assumptions hold (we cannot tell whether they do, but we suspect they do), GAs are optimal.

9.5 The power of GAs

The example of computing the maximum of $f(x) = -x^2 + 12x + 300$ is so simple that it would be easy to be lulled into thinking that GAs are toy models that are far from any useful level of modeling reality. However keep in mind that GAs have beaten human programmers in several instances. In one example the task is to produce short programs that can make copies of themselves. The shorter programs are deemed more successful and allowed to run longer. In this case a genetic algorithm has bested the best human programmers could do.

Another example perhaps closer to everyone's experience is the computer game of Pac Man. A huge percentage of people have played this game which involves a computer icon that can be steered in a maze that is visible from the top on a computer screen. Figure 9.7 shows a moment in the game. The Pac Man figure earns points by eating small pellets in the maze but can earn a lot more by eating one of four pills or an especially valuable moving fruit. The Pac Man must also avoid four monsters that chase it. The pills also have a special bonus in that for a short while after Pac man consumes one the monsters turn color and can be caught and captured.

This game was encoded as a genetic algorithm by Rosca[**?**]. The 'DNA' represented a program for directing the Pac Man figure. The program had some sensing ability to know where the nearest two monsters were and where the nearest pill and fruit were located. And it had movement instructions that would direct the motion toward or away from these objects. A population of random seed programs was created at the start of the algorithm but the algorithm did the rest. Random fit programs (based on how many points they achieved in a time interval) were selected for breeding. The crossover process interchanged pieces of program instructions, creating new programs.

If you have ever done any programming of any kind you might be suspicious at this point, wondering how the random interchange of instructions could even produce a program that would execute. The algorithm was greatly aided by using programs coded in the language LISP which has the great virtue in that crossover points that guarantee executable programs are easily selected.

Starting with different collections of random programs, the populations evolve to have different sets of behaviors. The results of one particularly interesting run are shown in Figure 9.7. The programs have evolved to wait by the pills until the monsters get close. Just before they strike, Pac Man eats the pill and runs them down. In the figure, three monsters have just been captured by this technique and Pac Man is in hot pursuit of a forth.

9.6 Why big brains?

At this point you have been through a whirlwind tour of the genetic revolution with an introduction to the modeling tool of genetic algorithms. Hopefully enough of the elements have been exposed to guide a tour of answers to the the basic question of this chapter: Why do humans have big brains?

- Bipedalism. Standing upright allows use of hands for complex manipulations
- Tool use. Tools place intellectual demands of the brain, leading to a spurt in complexity
- Language. The use of vocal symbols with syntax is a breakthrough that forces brain expansion.

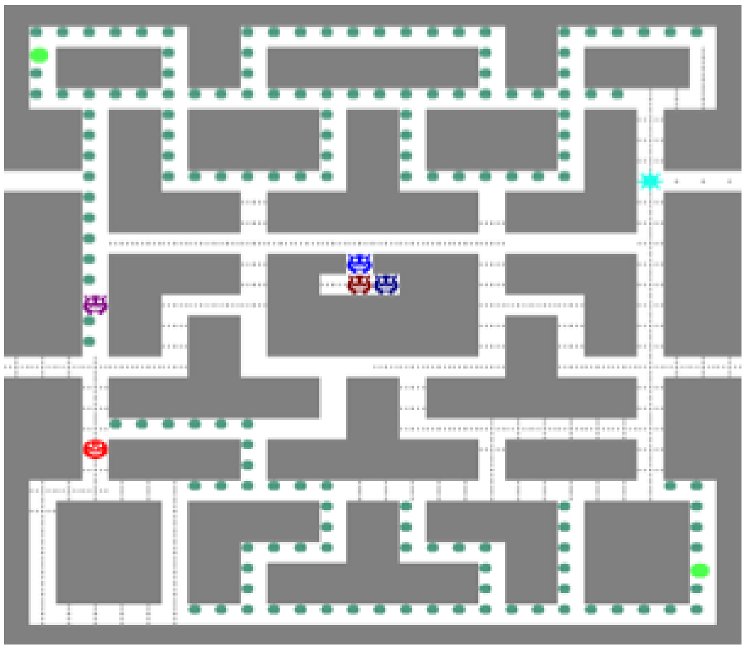

Figure 9.7: A moment in the trace of Rosca's genetic programming-based Pac-Man program. The Pac Man has waited in the south-west corner by moving back and forth to attract the four monsters. When they are very close, it eats the pill which temporarily bequeaths the power to capture monsters. Once captured they are sent to the central pit for a small amount of time. At the moment shown Pac man has captured three and is chasing a forth.

- Culture/social groups. The complex demands of successful interactions in large groups drives brain complexity.
- Software breakthrough. In trying to meet any or all of the above, brain architecture discovers new algorithms with dramatically improved performance.
- Sexual selection. female mates select for features that are produced by big brains.

In evaluating these alternate explanations, one can now be guided by an understanding of how the underlying genetic mechanisms work, as well as the fossil record. And of course the answer can easily be all of the above. It could be the case that each of the mechanisms outlined above conferred enough incremental improvement in fitness to be kept but there was a huge synergy in their ultimate combination. So the question being asked is closer to: Is there one item on the list that would stand out from the rest?

One further puzzle is the closest primate relatives the chimpanzees. Estimates vary, but roughly a third of primate DNA is thought to be used in brain construction and chimpanzee differs from our DNA by only a few tenths of a percentage point. Why did not chimps evolve big brains? As Willis points out [1], one major difference between chimps and us is that we are born with flexible skull plates that allow the brain to grow in size by a factor of three postnatally. Chimps by comparison have fixed skull plates so that their brains at birth are 80% of their final size.

Bipedalism was no doubt important in that if the hands are needed for basic locomotion they cannot easily be used for something else. Furthermore the development of hands with opposable thumbs in primates was accompanied with a corresponding area of the cortex. Nonetheless

Bibliography

[1] Christopher Willis. *The Runaway Brain: The Evolution of Human Uniqueness.* New York: Basic Books, 1993.