



Tutorial - Genetic Algorithms in Plain English

Introduction

The aim of this tutorial is to explain genetic algorithms sufficiently for you to be able to use them in your own projects. This is a stripped-down to-the-bare-essentials type of tutorial. I'm not going to go into a great deal of depth and I'm not going to scare those of you with math anxiety by throwing evil equations at you every few sentences. In fact, I'm not going to throw any nasty equations at you at all! Not in this particular tutorial anyway... <smile>

This tutorial is designed to be read through *twice*... so don't worry if little of it makes sense the first time you study it.

First, a Biology Lesson

Every organism has a set of rules, a blueprint so to speak, describing how that organism is built up from the tiny building blocks of life. These rules are encoded in the *genes* of an organism, which in turn are connected together into long strings called *chromosomes*. Each gene represents a specific trait of the organism, like eye colour or hair colour, and has several different settings. For example, the settings for a hair colour gene may be blonde, black or auburn. These genes and their settings are usually referred to as an organism's *genotype*. The physical expression of the genotype - the organism itself - is called the *phenotype*

When two organisms mate they share their genes. The resultant offspring may end up having half the genes from one parent and half from the other. This process is called *recombination*. Very occasionally a gene may be *mutated*. Normally this mutated gene will not affect the development of the phenotype but very occasionally it will be expressed in the organism as a completely new trait.

Life on earth has evolved to be as it is through the processes of natural selection, recombination and mutation. To illustrate how these processes work together to produce the diverse range of flora and fauna we share our planet with let me tell you a little story....

Once upon a time there lived a species of creatures called Hooters. Hooters had evolved entirely within the darkened confines of a vast cave system hidden deep in the bowels of a mountain range. They'd had an easy life, feeling and smelling around the damp cave walls for the algae they so loved to eat, oozing between rocks and, at mating time, listening intently for the hoots of other Hooters. There were no predators in the caves, it was just the Hooters, the algae and the occasional friendly slug, so the Hooters never had anything to fear (except for maybe the occasional bad tempered Hooter). An underground river flowed through the cave system and water continuously dripped down through the water table bringing with it the fresh nutrients the algae thrived on so there was always plenty to eat and drink. However, although Hooters could feel and hear well they never had any need for eyes in the pitch blackness of the caves and as a result were totally blind. This never seemed to concern any of the Hooters though and they all had a whale of a time munching away and hooting in the darkness.

Then one day an earthquake caused part of the cave system to collapse and for the first time in many millennia the Hooters felt the warmth of sunlight upon their skin and the soft springiness of moss beneath their feet. A

few daring Hooters tasted the moss and found that it was even better eating than the cave algae. "Ooooooooh!" they hooted between mouthfuls of moss and promptly got gobbled up by the marauding eagles who had flown in to see what all the commotion was about.

For a while it looked as though the Hooters may be hunted to extinction, for although they liked to eat the moss they could never tell if an eagle was flying above. Not only that, they couldn't even tell if they were concealed beneath a rock or not unless it was low enough to reach for with their feelers. Every day many Hooters would stumble out from the caves with the sweet smell of moss in their nostrils only to be swiftly carried away and eaten by an eagle. Their situation seemed grim indeed.

Fortunately, over the years, the population of Hooters had grown to be enormous in the safety of the caves and enough of them were surviving to mate - after all, an eagle can only eat so much. One day, a brood of Hooters was born that shared a mutated skin cell gene. This particular gene was responsible for the development of the skin cells on their foreheads. During the development of the baby Hooters, when their skin cells grew from the mutated gene instructions they were slightly light sensitive. Each new baby Hooter could sense if something was blocking the light to its forehead or not. When these little baby Hooters grew up into bigger Hooters and ventured into the light to eat the moss they could tell if something was swooping overhead or not. So these Hooters grew up to have a slightly better chance of survival than their totally blind cousins. And because they had a better chance of survival, they reproduced much more, therefore passing the new light sensitive skin cell gene to their offspring. After a very short while the population became dominated by the Hooters with this slight advantage.

Now let's zip a few thousand generations into the future. If you extrapolate this process over very many years and involving lots of tiny mutations occurring in the skin cell genes it's easy to imagine a process where one light sensitive cell may become a clump of light sensitive cells, and then how the interior cells of the clump may mutate to harden into a tiny lens shaped area, which would help to gather the light and focus it into one place. It's not too difficult to envision a mutation that gives rise to two of these light gathering areas thereby bestowing binocular vision upon the Hooters. This would be a huge advantage over their Cyclopsian cousins as the Hooters would now be able to judge distances accurately and have a greater field of view.

As you can see the processes of natural selection - survival of the fittest - and gene mutation have very powerful roles to play in the evolution of an organism. But how does recombination fit into the scheme of things? Well to show you that I need to tell about some other Hooters...

At around the same time the Hooters with the light sensitive cells were frolicking around in the moss and teasing the eagles, another brood of Hooters had been born who shared a mutated gene that affected their hooter. This mutation gave rise to a slightly bigger hooter than their cousins, and because it was bigger they could now hoot over longer distances. This turned out to be useful in the rapidly diminishing population because the Hooters with the bigger hooters could call out to potential mates situated far away. Not only that but the female Hooters began to show a slight preference to males with larger hooters. The upshot of this of course was that the better endowed males with larger hooters. The upshot of this of course was that the better endowed Hooters stood a much better chance of mating than any not so well off Hooters. Over a period of time, large hooters became prevalent in the population.

Then one fine day a female Hooter with the gene for light sensitive skin cells met a male Hooter with the gene for producing huge hooters. They fell in love, and shortly afterwards produced a brood of lovely baby Hooters. Now, because the babies chromosomes were a recombination of both parents chromosomes, some of the babies shared both the special genes and grew up not only to have light sensitive skin cells, but huge hooters too! These new offspring were extremely good at avoiding the eagles and reproducing so the process of evolution began to favour them and once again this new improved type of Hooter became dominant in the population.

And so on. And so on...

Genetic Algorithms are a way of solving problems by mimicking the same processes mother nature uses. They use the same combination of selection, recombination and mutation to evolve a solution to a problem. Neat huh? Turn the page to find out exactly how it's done.

The Genetic Algorithm - a brief overview

Before you can use a genetic algorithm to solve a problem, a way must be found of *encoding* any potential solution to the problem. This could be as a string of real numbers or, as is more typically the case, a binary bit string. I will refer to this bit string from now on as the chromosome. A typical chromosome may look like this:

10010101110101001010011101101110111111101

(Don't worry if none of this is making sense to you at the moment, it will all start to become clear shortly. For now, just relax and go with the flow.)

At the beginning of a run of a genetic algorithm a large population of *random* chromosomes is created. Each one, when decoded will represent a different solution to the problem at hand. Let's say there are N chromosomes in the initial population. Then, the following steps are repeated until a solution is found

1. Test each chromosome to see how good it is at solving the problem at hand and assign *fitness score* accordingly. The fitness score is a measure of how good that chromosome is at solving the problem to hand.
2. Select two members from the current population. The chance of being selected is proportional to the chromosomes fitness. *Roulette wheel* selection is a commonly used method.
3. Dependent on the *crossover rate* crossover the bits from each chosen chromosome at a randomly chosen point.
4. Step through the chosen chromosomes bits and flip dependent on the *mutation rate*.
5. Repeat step 2, 3, 4 until a new population of N members has been created.

Tell me about Roulette Wheel selection

This is a way of choosing members from the population of chromosomes in a way that is proportional to their fitness. It does not guarantee that the fittest member goes through to the next generation, merely that it has a very good chance of doing so. It works like this:

Imagine that the population's total fitness score is represented by a pie chart, or roulette wheel. Now you assign a slice of the wheel to each member of the population. The size of the slice is proportional to that chromosomes fitness score. i.e. the fitter a member is the bigger the slice of pie it gets. Now, to choose a chromosome all you have to do is spin the ball and grab the chromosome at the point it stops.

What's the Crossover Rate?

This is simply the chance that two chromosomes will swap their bits. A good value for this is around 0.7. Crossover is performed by selecting a random gene along the length of the chromosomes and swapping all the genes after that point.

e.g. Given two chromosomes

10001001110010010
01010001001000011

Choose a random bit along the length, say at position 9, and swap all the bits after that point so

the above become:

10001001101000011
01010001010010010

What's the Mutation Rate?

This is the chance that a bit within a chromosome will be flipped (0 becomes 1, 1 becomes 0). This is usually a very low value for binary encoded genes, say 0.001

So whenever chromosomes are chosen from the population the algorithm first checks to see if crossover should be applied and then the algorithm iterates down the length of each chromosome mutating the bits if applicable.

From Theory to Practice

To hammer home the theory you've just learnt let's look at a simple problem:

*Given the digits 0 through 9 and the operators +, -, * and /, find a sequence that will represent a given target number. The operators will be applied sequentially from left to right as you read.*

So, given the target number 23, the sequence $6+5*4/2+1$ would be one possible solution.

If 75.5 is the chosen number then $5/2+9*7-5$ would be a possible solution.

Please make sure you understand the problem before moving on. I know it's a little contrived but I've used it because it's very simple.

Stage 1: Encoding

First we need to encode a possible solution as a string of bits... a chromosome. So how do we do this? Well, first we need to represent all the different characters available to the solution... that is 0 through 9 and +, -, * and /. This will represent a gene. Each chromosome will be made up of several genes.

Four bits are required to represent the range of characters used:

0: 0000
1: 0001
2: 0010
3: 0011
4: 0100
5: 0101
6: 0110
7: 0111
8: 1000
9: 1001
+: 1010
-: 1011
*: 1100
/: 1101

The above show all the different genes required to encode the problem as described. The possible genes **1110** & **1111** will remain unused and will be ignored by the algorithm if encountered.

So now you can see that the solution mentioned above for 23, '6+5*4/2+1' would be represented by nine genes like so:

0110 1010 0101 1100 0100 1101 0010 1010 0001
6 + 5 * 4 / 2 + 1

These genes are all strung together to form the chromosome:

011010100101110001001101001010100001

A Quick Word about Decoding

Because the algorithm deals with random arrangements of bits it is often going to come across a string of bits like this:

0010001010101110101101110010

Decoded, these bits represent:

0010 0010 1010 1110 1011 0111 0010
2 2 + n/a - 7 2

Which is meaningless in the context of this problem! Therefore, when decoding, the algorithm will just ignore any genes which don't conform to the expected pattern of: number -> operator -> number -> operator ...and so on. With this in mind the above 'nonsense' chromosome is read (and tested) as:

2 + 7

Stage 2: Deciding on a Fitness Function

This can be the most difficult part of the algorithm to figure out. It really depends on what problem you are trying to solve but the general idea is to give a higher fitness score the closer a chromosome comes to solving the problem. With regards to the simple project I'm describing here, a fitness score can be assigned that's inversely proportional to the difference between the solution and the value a decoded chromosome represents.

If we assume the target number for the remainder of the tutorial is 42, the chromosome mentioned above

011010100101110001001101001010100001

has a fitness score of $1/(42-23)$ or $1/19$.

As it stands, if a solution is found, a divide by zero error would occur as the fitness would be $1/(42-42)$. This is not a problem however as we have found what we were looking for... a solution. Therefore a test can be made for this occurrence and the algorithm halted accordingly.

Stage 3: Getting down to business

First, please read this tutorial again.

If you now feel you understand enough to solve this problem I would recommend trying to code the genetic algorithm yourself. There is no better way of learning. If, however, you are still confused, I have already prepared some simple code which you can find [here](#). Please tinker around with the mutation rate, crossover rate, size of chromosome etc to get a feel for how each parameter effects the algorithm. Hopefully the code should be documented well enough for you to follow what is going on! If not please email me and I'll try to improve the commenting.

Note: The code given will parse a chromosome bit string into the values we have discussed and

it will attempt to find a solution which uses *all* the valid symbols it has found. Therefore if the target is 42, $+ 6 * 7 / 2$ would not give a positive result even though the first four symbols (" + 6 * 7") do give a valid solution.

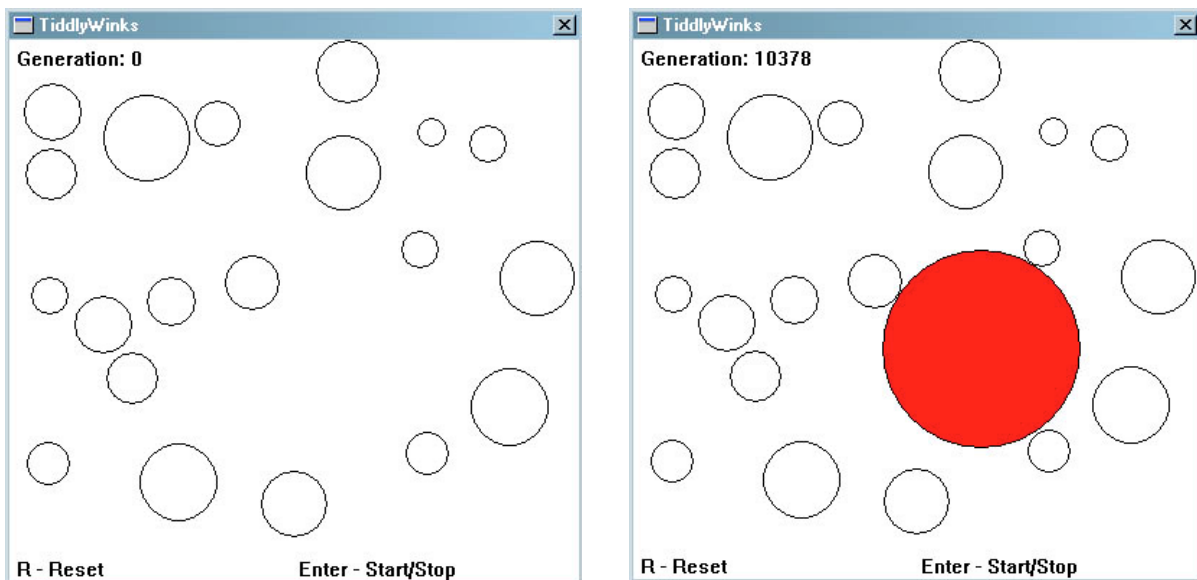
Last Words

I hope this tutorial has helped you get to grips with the basics of genetic algorithms. Please note that I have only covered the very basics here. If you have found genetic algorithms interesting then there is much more for you to learn. There are different selection techniques to use, different crossover and mutation operators to try and more esoteric stuff like fitness sharing and speciation to fool around with. All or some of these techniques will improve the performance of your genetic algorithms considerably.

Stuff to Try

If you have succeeded in coding a genetic algorithm to solve the problem given in the tutorial, try having a go at the following more difficult problem:

Given an area that has a number of non-overlapping disks scattered about its surface as shown below left.



Use a genetic algorithm to find the disk of largest radius which may be placed amongst these disks without overlapping any of them. See figure above right.