Genetic Algorithm(s)

A. Godunov

1. Basics: GA as numerical optimization algorithm
2. Genetic Algorithm
3. Improving the algorithm

1

Part 1:

Basics

2

Books – so many!

An Introduction to Genetic Algorithms for Scientists and Engineers
David A Coley
An Introduction to Genetic Algorithms
Melanie Mitchell
Genetic Algorithms in Search, Optimization & Machine Learning
David E. Goldberg
Genetic Algorithms and Machine Learning for Programmers
Frances Buontempo
Hands-On Genetic Algorithms with Python
Evolutionary Optimization Algorithms
Dan Simon
Practical Genetic Algorithms
Second Edition
Randy L. Haupt
Sue Ellen Haupt
Introduction to Stochastic Search and Optimization
James C. Spall
Algorithms for Optimization
Mykel J. Kochenderfer and Tim A. Wheeler

3

What is GA?

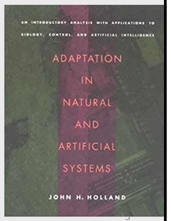
Genetic algorithms (GA) are numerical optimization algorithms motivated by natural selection and natural genetics.

The method is a general one, capable of being applied to an extremely wide range of problems.

4

History of the GA

- From the very early years of computers, computer scientists have had visions of systems that mimicked one or more of the attributes of life.
- The idea of using a population of solutions to solve practical engineering optimization problems was considered several times during the 1950's and 1960's.
- GA was created by John Holland "Adaptation in Natural and Artificial Systems" 1975



5

Applications (general areas)

- Physics and Chemistry
- Mathematics and Computer Science
- Biological Sciences and Bioinformatics
- Medicine
- Industry, Management and Engineering
- Finance and Economics
- Earth Sciences
- Social Sciences
- and more …

6

Applications (some examples)

- VLSI (very large scale integration) electronic chip layouts
- spacecraft trajectories
- robotics
- water networks
- the architectural aspects of building design
- Factory floor scheduling scheduling (Volvo, Deere, …)
- facial recognition
- Crashworthy car design (GM)
- Data mining

A typical algorithm might consist of the following:

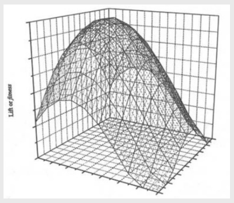
1. a number, or population, of guesses of the solution to the problem;
2. a way of calculating how good or bad the individual solutions within the population are;
3. a method for mixing fragments of the better solutions to form new, on average even better solutions; and
4. a mutation operator to avoid permanent loss of diversity within the solutions.

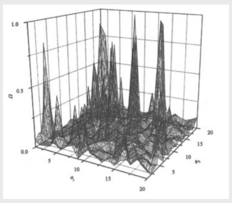
Search spaces

In a numerical search or optimization problem, a list, quite possibly of infinite length, of possible solutions is being searched in order to locate the solution that best describes the problem at hand

Example: maximize the lift generated by an airplane's wing.

If there were only two of these adjustable parameters, a and b, one could try a large number of combinations, calculate the lift generated by each design and produce a surface plot with a, b





Search spaces (cont.)

For more complex problems, with more than two unknowns, the situation becomes harder to visualize!

The concept of a search space is valid as long as some measure of distance between solutions can be defined and each solution can be assigned a measure of success, or fitness, within the problem.

Better performing, or fitter, solutions will then occupy the peaks within the search space (or fitness landscape) and poorer solutions the valleys.

Such spaces or landscapes can be of surprisingly complex topography. Even for simple problems, there can be numerous peaks of varying heights, separated from each other by valleys on all scales.

The highest peak is usually referred to as the global maximum or global optimum, the lesser peaks as local maxima or local optima.

A measure of success

For more complex problems, with more than two unknowns, the situation becomes harder to visualize!

The concept of a search space is valid as long as some measure of distance between solutions can be defined and each solution can be assigned a measure of success, or fitness, within the problem.

Better performing, or fitter, solutions will then occupy the peaks within the search space (or fitness landscape) and poorer solutions the valleys.

Such spaces or landscapes can be of surprisingly complex topography. Even for simple problems, there can be numerous peaks of varying heights, separated from each other by valleys on all scales.

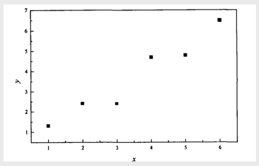
The highest peak is usually referred to as the global maximum or global optimum, the lesser peaks as local maxima or local optima.

The goal of optimization

- For most search problems, the goal is the accurate identification of the global optimum
- In some situations, for example real-time control, the identification of any point above a certain value of fitness might be acceptable
- For other problems, for example, in architectural design, the identification of a large number of highly fit, yet distant and therefore distinct, solutions (designs) might be required

A simple example

Consider the experimental data



Assume that x and y are connected as $y_j = mx_j + c$

But what values should be given to m and c?

If there is reason to believe that $y = 0$ when $x = 0$ (i.e. the line passes through the origin) then $c = 0$ and m is the only adjustable parameter (or unknown). Let y_j are experimental points, then we can use a least-squares estimation $\Omega = \sum_{j=1}^{n} \left(y_j - y_j\right)^2$

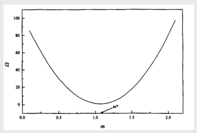
13

13

A simple example (cont.)

We need to find such m that $\Omega = \sum_{j=1}^{n} \left(y_j - my_j\right)^2$ is a minimum.

One way to do this is to use a computer to calculate Ω over a fine grid of values of m. Then simply choose the m which generates the lowest value of Ω.



This approach, of estimating an unknown parameter, or parameters, by simply solving the problem for a very large number of values of the unknowns is called an *enumerative search*.

It is only really useful if there are relatively few unknown parameters and one can estimate Ω rapidly.

14

14

From simple to more complicated

Consider a problem in which there are ten unknowns, each of which are required to an accuracy of one percent

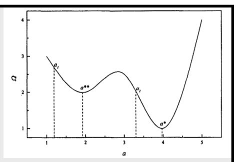
It will require 100^{10}, or 10^{20}, estimations! (years to complete)

Given that ten is not a very large number of unknowns, one percent not a very demanding level of accuracy clearly there is a need to find a better approach.

15

15

More complications



A more complex one-dimensional search space with both a global and a local minimum.

If either the direct search algorithm is used, the final estimate of the parameter a will depend on where in the search space the algorithm was started. Making the initial guess at $a = a_2$, will indeed lead to the correct (or global) minimum, a^*. However, if $a = a_1$ is used then only a^{**} will be reached (a local minimum).

This highlights a serious problem. If the results produced by a search algorithm depend on the starting point, then there will be little confidence in the answers generated.

One way around this problem would be to start the problem from a series of points and then assume that the true global minimum lies at the lowest minimum identified

16

16

Part 2:

Genetic Algorithm

17

Key ideas

Rather than starting from a single point (or guess) within the search space, GA is initialized with a *population* of guesses.

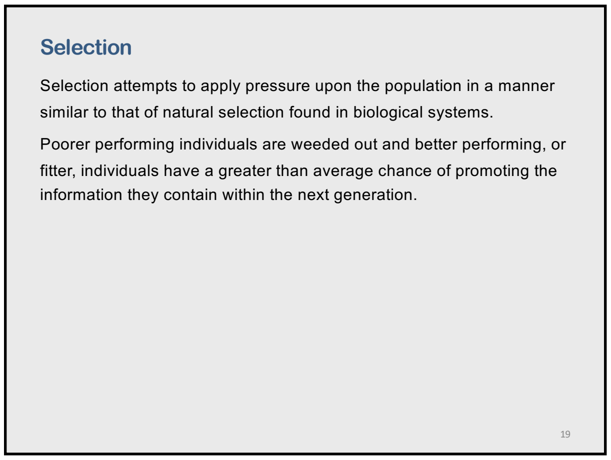
These are usually random and will be spread throughout the search space.

A typical algorithm then uses three operators, *selection*, *crossover* and *mutation* (chosen in part by analogy with the natural world) to direct the population (over a series of time steps or generations) towards convergence at the global optimum.

Typically, these initial guesses are held as binary encodings (or strings) of the true variables, although an increasing number of GA use "real-valued" (i.e. base-10) encodings.

18

18

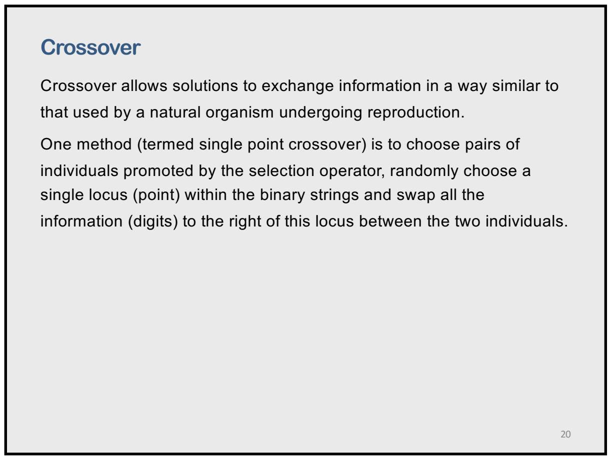


Selection

Selection attempts to apply pressure upon the population in a manner similar to that of natural selection found in biological systems.

Poorer performing individuals are weeded out and better performing, or fitter, individuals have a greater than average chance of promoting the information they contain within the next generation.

19

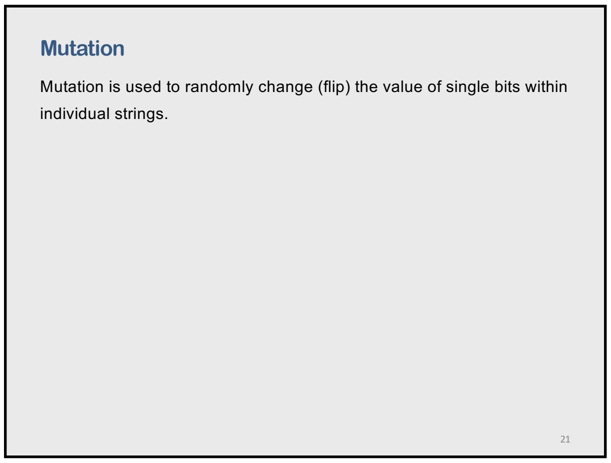


Crossover

Crossover allows solutions to exchange information in a way similar to that used by a natural organism undergoing reproduction.

One method (termed single point crossover) is to choose pairs of individuals promoted by the selection operator, randomly choose a single locus (point) within the binary strings and swap all the information (digits) to the right of this locus between the two individuals.

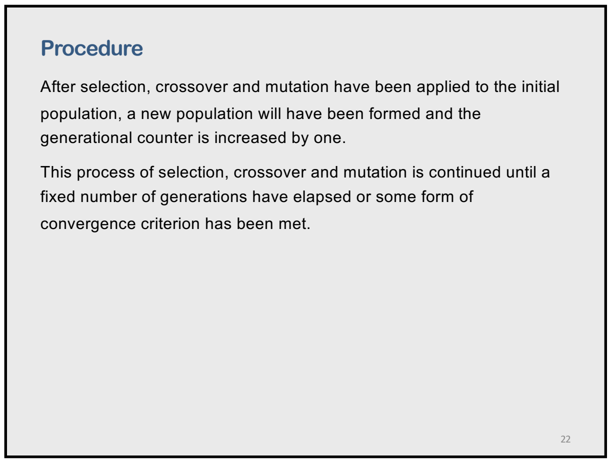
20



Mutation

Mutation is used to randomly change (flip) the value of single bits within individual strings.

21

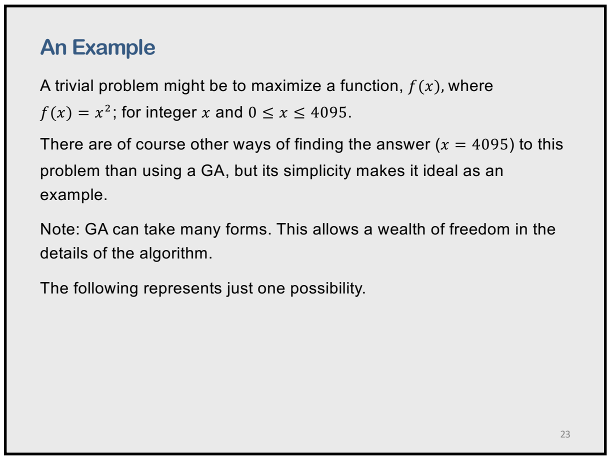


Procedure

After selection, crossover and mutation have been applied to the initial population, a new population will have been formed and the generational counter is increased by one.

This process of selection, crossover and mutation is continued until a fixed number of generations have elapsed or some form of convergence criterion has been met.

22



An Example

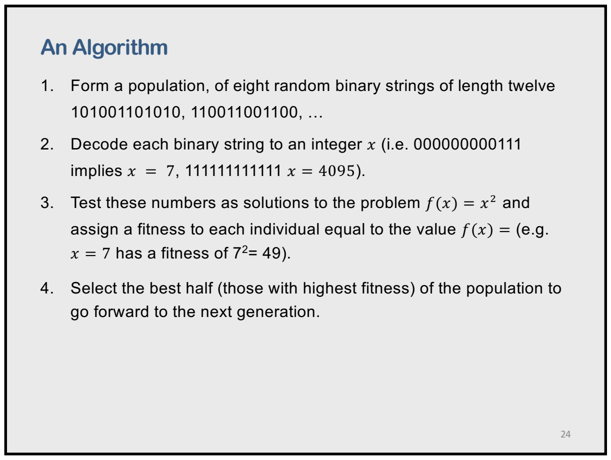
A trivial problem might be to maximize a function, $f(x)$, where $f(x) = x^2$; for integer x and $0 \leq x \leq 4095$.

There are of course other ways of finding the answer ($x = 4095$) to this problem than using a GA, but its simplicity makes it ideal as an example.

Note: GA can take many forms. This allows a wealth of freedom in the details of the algorithm.

The following represents just one possibility.

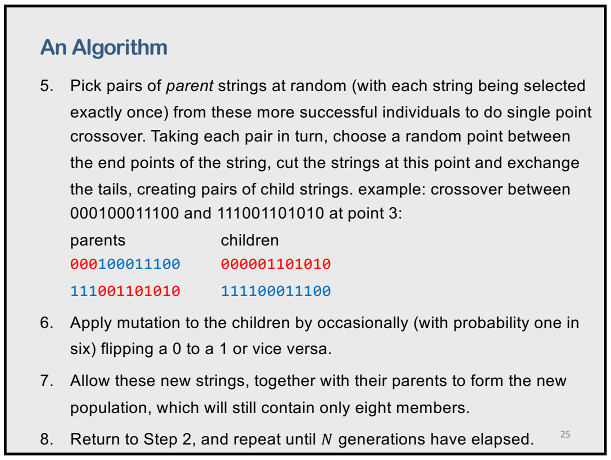
23



An Algorithm

1. Form a population, of eight random binary strings of length twelve 101001101010, 110011001100, …
2. Decode each binary string to an integer x (i.e. 000000000111 implies $x = 7$, 111111111111 $x = 4095$).
3. Test these numbers as solutions to the problem $f(x) = x^2$ and assign a fitness to each individual equal to the value $f(x)$ = (e.g. $x = 7$ has a fitness of 7^2= 49).
4. Select the best half (those with highest fitness) of the population to go forward to the next generation.

24



An Algorithm

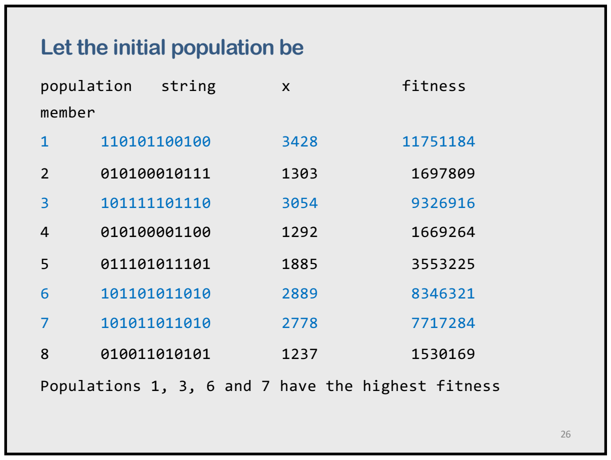
5. Pick pairs of *parent* strings at random (with each string being selected exactly once) from these more successful individuals to do single point crossover. Taking each pair in turn, choose a random point between the end points of the string, cut the strings at this point and exchange the tails, creating pairs of child strings. example: crossover between 000100011100 and 111001101010 at point 3:

parents	children
000100011100	000001101010
111001101010	111100011100

6. Apply mutation to the children by occasionally (with probability one in six) flipping a 0 to a 1 or vice versa.
7. Allow these new strings, together with their parents to form the new population, which will still contain only eight members.
8. Return to Step 2, and repeat until N generations have elapsed.

25

25



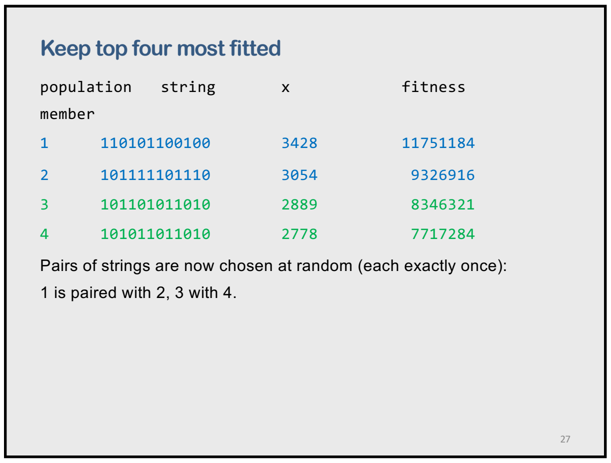
Let the initial population be

population member	string	x	fitness
1	110101100100	3428	11751184
2	010100010111	1303	1697809
3	101111101110	3054	9326916
4	010100001100	1292	1669264
5	011101011101	1885	3553225
6	101101011010	2889	8346321
7	101011011010	2778	7717284
8	010011010101	1237	1530169

Populations 1, 3, 6 and 7 have the highest fitness

26

26



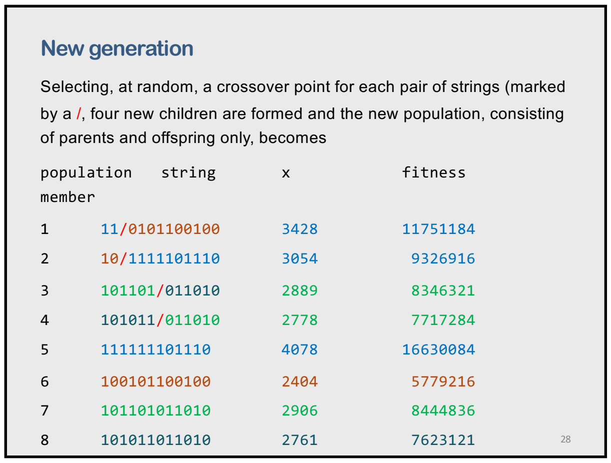
Keep top four most fitted

population member	string	x	fitness
1	110101100100	3428	11751184
2	101111101110	3054	9326916
3	101101011010	2889	8346321
4	101011011010	2778	7717284

Pairs of strings are now chosen at random (each exactly once):
1 is paired with 2, 3 with 4.

27

27



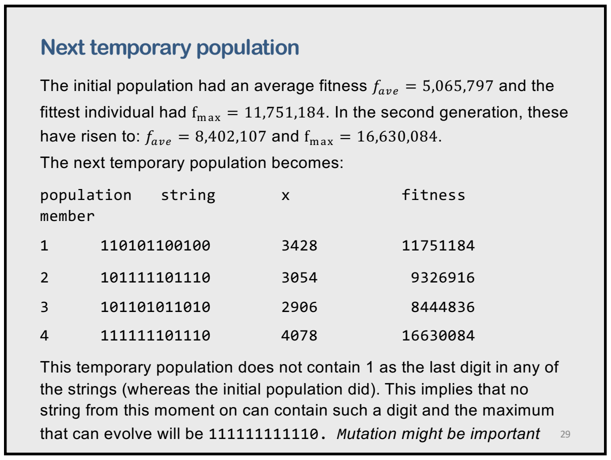
New generation

Selecting, at random, a crossover point for each pair of strings (marked by a /, four new children are formed and the new population, consisting of parents and offspring only, becomes

population member	string	x	fitness
1	11/0101100100	3428	11751184
2	10/1111101110	3054	9326916
3	101101/011010	2889	8346321
4	101011/011010	2778	7717284
5	111111101110	4078	16630084
6	100101100100	2404	5779216
7	101101011010	2906	8444836
8	101011011010	2761	7623121

28

28



Next temporary population

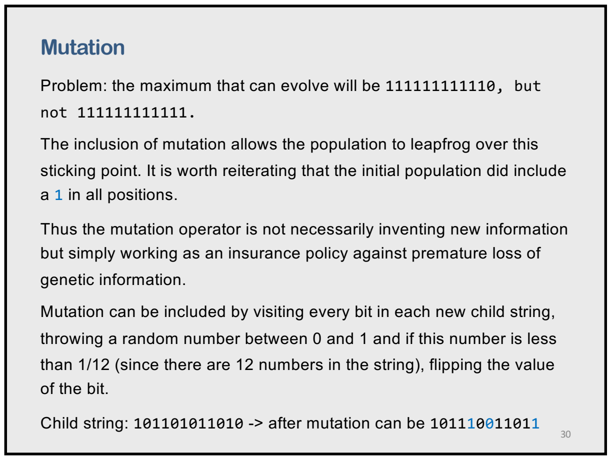
The initial population had an average fitness $f_{ave} = 5{,}065{,}797$ and the fittest individual had $\mathrm{f}_{\max} = 11{,}751{,}184$. In the second generation, these have risen to: $f_{ave} = 8{,}402{,}107$ and $\mathrm{f}_{\max} = 16{,}630{,}084$.
The next temporary population becomes:

population member	string	x	fitness
1	110101100100	3428	11751184
2	101111101110	3054	9326916
3	101101011010	2906	8444836
4	111111101110	4078	16630084

This temporary population does not contain 1 as the last digit in any of the strings (whereas the initial population did). This implies that no string from this moment on can contain such a digit and the maximum that can evolve will be 111111111110. *Mutation might be important*

29

29



Mutation

Problem: the maximum that can evolve will be 111111111110, but not 111111111111.

The inclusion of mutation allows the population to leapfrog over this sticking point. It is worth reiterating that the initial population did include a 1 in all positions.

Thus the mutation operator is not necessarily inventing new information but simply working as an insurance policy against premature loss of genetic information.

Mutation can be included by visiting every bit in each new child string, throwing a random number between 0 and 1 and if this number is less than 1/12 (since there are 12 numbers in the string), flipping the value of the bit.

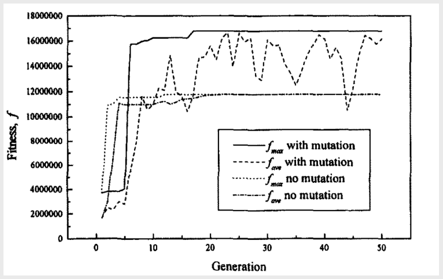
Child string: 101101011010 -> after mutation can be 101110011011

30

30

Calculations

The evolution of the population. The fitness of the best performing individual f_{max} is seen to improve with generation as is the average fitness of the population f_{ave}. Without mutation the lack of a I in all positions limits the final solution.



31

31

Summary

The three central operators behind the method are *selection, crossover* and *mutation*.

Using these operators a very simple GA has been constructed and applied to a trivial problem.

Although a genetic algorithm has now been successfully constructed and applied to a simple problem, it is obvious that many questions remain.

In particular, how are problems with more than one unknown dealt with, and how are problems with real (or complex) valued parameters to be tackled?

32

32

Part 3:

Improving the Algorithm

33

Some questions

1. How will the algorithm perform across a wider range of problems?
2. How are non-integer unknowns tackled?
3. How are problems of more than one unknown dealt with?
4. Are there better ways to define the selection operator that distinguishes between good and very good solutions?

34

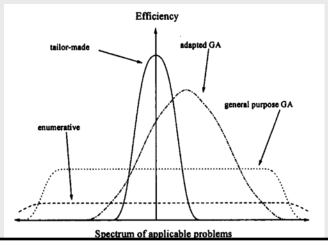
34

Robustness

The more robust the algorithm the greater the range of problems it can be applied to.

A tailor-made method such as a traditional calculus based algorithm might be highly efficient for some problems, but will fail on others.

GAs are naturally robust and therefore effective across a wide range of problems.



35

35

Non-integer Unknowns

From integer to float such as -3.62, or $1.23 * 10^{-4}$.

There are many ways of doing this; however the most common is by a linear mapping between the real numbers and a binary representation of fixed length.

36

36

Multiparameter Problems

Extending the representation to problems with more than one unknown proves to be particularly simple.

The M unknowns are each represented as sub-strings of length l. These sub-strings are then concatenated (joined together) to form an individual population member of length L, where:

$$L = \sum_{j=1}^{M} l_j$$

For example, given a problem with two unknowns a and b, then if $a = 10110$ and $b = 11000$ for one guess at the solution, then by concatenation, the genotype is a $a \oplus b = 1011011000$.

Two things: 1) there is no need for the sub-strings used to represent a and b to be of the same length; this allows varying degrees of accuracy to be assigned to different parameters; 2) in general, the crossover cut point will not be between parameters but within a parameter.

37

37

Mutation

In the natural world, several processes can cause mutation, the simplest being an error during replication.

With a simple binary representation, mutation is particularly easy to implement. With each new generation the whole population is swept, with every bit position in every string visited and very occasionally a 1 is flipped to a 0 or vice versa (e.g. with probability $p_m \approx 1/L$).

However, just like everything else about GA, the correct setting for p_m will be problem dependent.

Example: $p_m \approx 1/(N\sqrt{L})$ where N is the population size

Observation: too low rate is likely to be less disastrous than too high rate for most problems.

38

38

Selection

Thus far, the selection operator has been particularly simple: the best 50% are selected to reproduce and the rest thrown away.

This is a practical method but not the most common.

A more common selection operator is fitness-proportional, or roulette wheel, selection. With this approach the probability of selection is proportional to an individual's fitness.

39

39

Elitism

Fitness-proportional selection does not guarantee the selection of any particular individual, including the fittest. Unless the fittest individual is much, much fitter than any other it will occasionally not be selected. To not be selected is to die.

Thus with fitness-proportional selection the best solution to the problem discovered so far can be regularly thrown away.

Although it appears counterproductive, this can be advantageous for some problems because it *slows* the algorithm, allowing it to explore more of the search space before convergence.

For many applications the search speed can be greatly improved by not losing the best, or elite, member between generations.

Ensuring the propagation of the elite member is termed elitism and requires that not only is the elite member selected, but a copy of it does not become disrupted by crossover or mutation.

40

40

Books – again

An Introduction to Genetic Algorithms for Scientists and Engineers
David A Coley
An Introduction to Genetic Algorithms
Melanie Mitchell
Genetic Algorithms in Search, Optimization & Machine Learning
David E. Goldberg
Genetic Algorithms and Machine Learning for Programmers
Frances Buontempo
Hands-On Genetic Algorithms with Python
Evolutionary Optimization Algorithms
Dan Simon
Practical Genetic Algorithms
Randy L. Haupt, Sue Ellen Haupt
Introduction to Stochastic Search and Optimization
James C. Spall
Algorithms for Optimization
Mykel J. Kochenderfer and Tim A. Wheeler

41

41