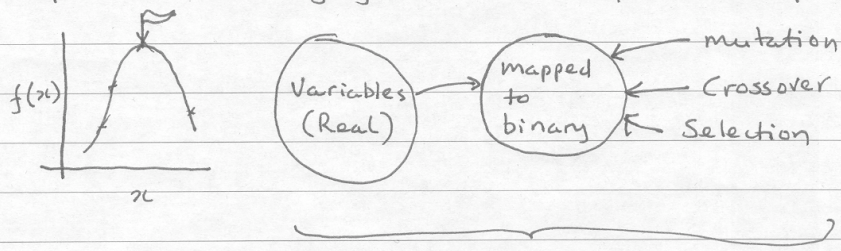
Genetic Algorithms - Lecture 3

- Recap: We are trying to solve an optimisation problem.



- Mutation

  In life, mutation is a small change (usually) in the genetic sequence of DNA.

  When the guessed values of x_i, optimised using selection and crossover, are close to the actual optimum value, leads to stagnation and mutation can help to get nearer the right answer.

- Bitwise mutation 1 0 0 1 0 0 0 1 0 1

  probability of mutation p_μ which is a small fraction (0.01 → 0.1)

  We the generate a random number (0 → 1) and if p_μ > random number, change selected bit from 0 to 1 or 1 to 0 depending on its initial value.

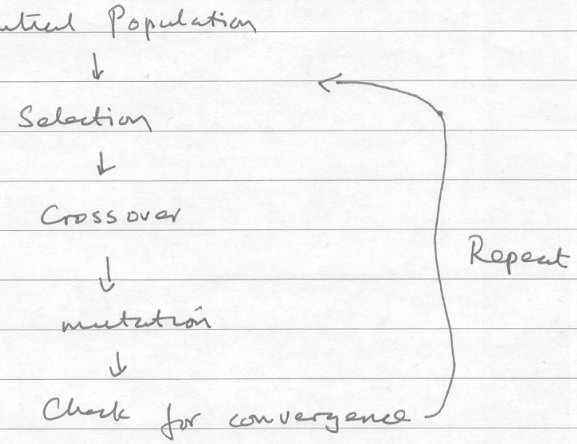
  Because mutation leads to a SMALL change, it can access the optimum solution even when selection and crossover have reached close to the optimum.

  Mutation can help avoid local maxima.

  Problem with bitwise mutation (also known as jump mutation) is that sometimes there can be a very large change if the sign first bit in the string mutates. Therefore, creep mutation

- Creep Mutation

  Convert string 10001 to real, perturb it slightly and convert back to string
  This avoids huge mutations

- 

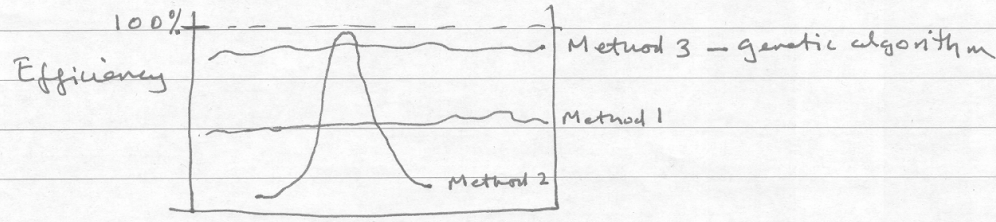
When do we decide the calculation is finished?
Practical: if 100 and 150 iterations give similar results then stop.
Alternatively: If the maximum fitness f_{max} in the population, scaled with average fitness $\bar{f}$ i.e., $\frac{f^{max}}{\bar{f}} \simeq 1$ then we say that there is convergence.

Problem is that one may have reached a local optimum. Therefore alter the mutation probability to a larger value and check whether different optimum reached.

* ← Simple Genetic Algorithms end here → *

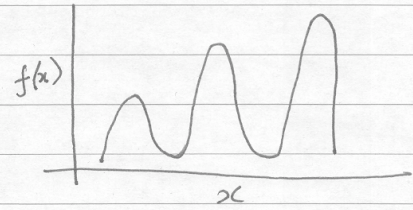
See book by David Goldberg

Why genetic algorithms, and not other optimisation methods ?



∴ Genetic algorithm is said to be robust, can be generally applied.

- Suppose we wish to discover all maxima in a function rather than the highest peak. This is a multimodal problem.



If all three peaks have the same height, then genetic algorithm picks one due to "genetic drift", especially when mutations are not large.

→ Sharing and niching

Define a "tax" which reduces fitness if guesses are similar.

1 0 0 1 0 1 0
1 0 0 0 0 0 0
↑ ↑
only two bits are different

Hamming distance between two strings is a measure of similarity.

with corresponding real numbers

	x_1	x_2	x_3
i	10	5	2
j	9	3	1

define Euclidean distance to check for similarity

$$\delta_{i,j} = \sqrt{\left(x_1^i - x_1^j\right)^2 + \left(x_2^i - x_2^j\right)^2 + \left(x_3^i - x_3^j\right)^2 \ldots}$$

Best to normalise the variables for this purpose

For n variables, n-dimensional hyperspace with "volume" kr^n where r is distance

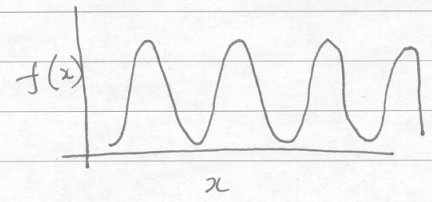
For every variable we have x_i^{upper} and x_i^{lower}

For those limits,

$$\delta_{i,j}^{max} = \sqrt{\sum_i \left(x_i^{upper} - x_i^{lower}\right)^2}$$ i = number of variables

take $\frac{1}{2}\delta_{i,j}^{max} = \mathcal{R}$, the radius of hyper volume

Total search space is $k\mathcal{R}^n$ where n = number of variables



We need to know at the outset the number of peaks $m = 4$

Divide the total hyperspace into m parts, each of radius ~~tangent volume~~ σ_{share} assuming equal spacing of peaks

$$k\mathcal{R}^n = mk\sigma_{share}^n$$