Tutorial

A worked problem-solving exercise

In many engineering problems, we need to solve complicated equations (e.g. partial differential equations). In a general simplified notation, an equation may be written as

$f(x)=c$

where c is a real valued constant vector, x is a real valued unknown vector and f is a complicated multimodal function, sometimes not even differentiable. The task is to find x that satisfies the equation.

Design an evolutionary algorithm to do this!

 We know how to solve function optimization by evolutionary algorithms. Can we also solve equations also?

 To design an evolutionary algorithm means to design the following *for the given problem*:

(a) the genotypic representation of an individual

(b) the fitness evaluation function for an individual

(c) evolutionary operators
- crossover (recombination)
- mutation

(d) selection scheme

(e) stopping criteria

While doing this, you should also:
- explain & justify all your design choices!
- comment on possible alternatives
- comment on possible problems and weaknesses of your solution & give suggestions for potential further improvement.

Some terminology:

Hamming Cliffs

•Hamming distance = number of bit-wise differences

E.g.

a) the Hamming distance of 1000 and 0111 is 4
b) the Hamming distance of 0000 and 1000 is 1

In a), the distance between phenotypes is 1 (1000=8, 0111=7) but the Hamming distance is 4

In b), the distance between phenotypes is 8 (0000=0, 1000=8) but the Hamming distance is 1.

So, a one-bit mutation can make a large (or a small) jump; a multi-bit mutation can make a small (or large) jump. This is known as the hamming cliff problem.

•Gray coding: Encode the natural numbers differently, such that neighbors always have a hamming distance one:

```
000 = 0
001 = 1
011 = 2
010 = 3
110 = 4
111 = 5
101 = 6
100 = 7
```

(initially called ‘reflected binary coding’)

Evolutionary operations:

c) Remember: Even if a solution can be represented, it doesn't mean that it is also reachable. Evolutionary operators need to be designed to allow nonzero probabilities of visiting all search space.

Mutation: bell-shaped distributions can be used

( ) Gaussian mutation
( ) Cauchy mutation
( ) Other? – what?

d) a mix of Gaussian and Cauchy mutations gives nicely distributed mixes of small and large mutations - see Lecture 1 (the guest lecture)

e) self-adaptation
   a. improves performance by controlling the spread
   b. reduces user-defined parameters
   c. if we adopt it, remember we need to revise our representation because each individual now needs to contain not just the actual function-values but also the adaptation parameters.

f) also, we must take care to do not let the adaptation parameter go near zero

 Stopping criteria

( ) set a maximum number of function evaluations
( ) stop when the improvement in fitness values over several generations is smaller than a pre-defined threshold.

 Anything else?

 Hmmm… Is it good if we achieve zero error?

a. What if the data was noisy?