Genetic Algorithms and Genetic Programming

Lecture **3**: (2/10/09)

The Canonical Genetic Algorithm

School of informatics

Michael Herrmann

michael.herrmann@ed.ac.uk, phone: 0131 6 517177, Informatics Forum 1.42

Overview

1. Introduction: History
2. The genetic code
3. The canonical genetic algorithm
4. Examples & practical issues
5. The schema theorem

Now: Implications for GA from natural evolution

Selection – Survival of the Fittest?

When individuals exist in populations, they **compete** for resources.

The fitter ones live, the less fit die.

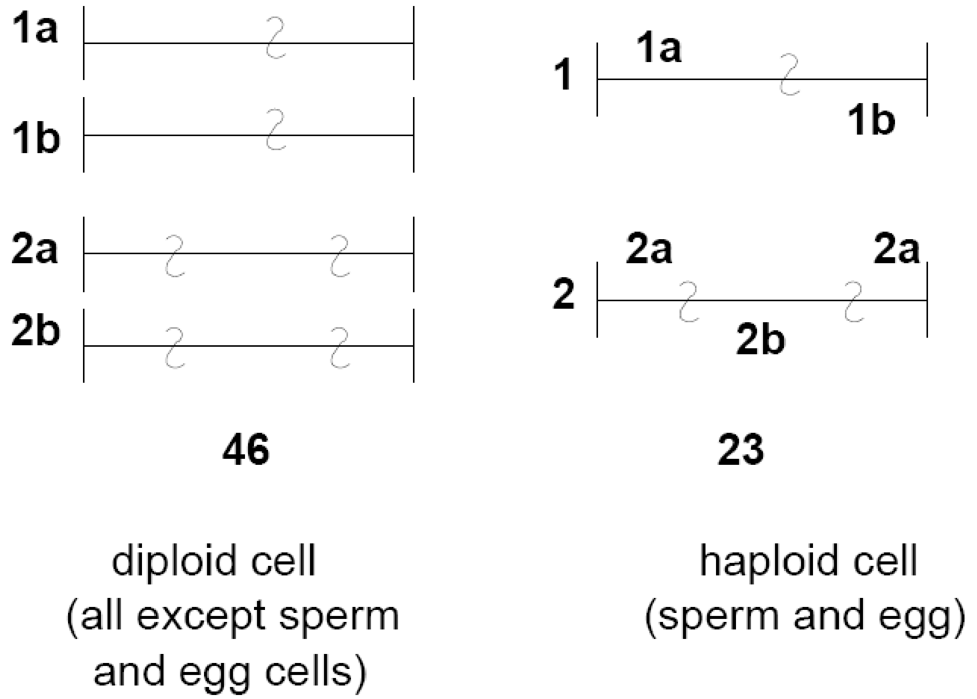
The survivors choose mates and produce offspring. The offspring share similarities with, but are not direct copies of, the parents:

- children inherit traits from parents
- offspring vary in their physical properties and behaviour

How can we explain this?

- sexual reproduction
- mutation
- crossover
- inversion

Crossover



Every haploid cell is a result of **different** recombinations of the 46 chromosomes.

When sperm and egg fuse, the 23 from the male and the 23 from the female are simply collected together to form a new collection of 46.

Mutation

During crossover, sometimes copying errors are made, with low probability:

. . . CGTATTCATGG . . .
. . . CGTA**C**TCATGG . . .

Also, sometimes strands of DNA become detached and flip end-over-end before reattaching – **inversion**:

. . . CGTATTCATGG . . .
. . . CGTA**ACTT**TGG . . .

So different amino acids are coded for. Crossover can be disruptive – it favours smaller units of inheritance (Dawkins).

Some (very large) parts of DNA seem to be there to protect genes from crossover (junk DNA, introns, bloat). Free radicals can also disrupt DNA.

Selection – Fitness Landscapes

Selection governs which organisms live long enough to pass on their genes.

Selection pressure defines a fitness landscape which favours one type of creature over others. E.g. in landscapes that provide sparse food, creatures that store fat efficiently are more likely to survive.

$\longrightarrow$ optimisation

Populations can also adapt to changes in their environment. E.g. if populations start farming rather than hunting/gathering they develop enzymes that can digest grains.

$\longrightarrow$ adaptation

Genetics and Evolution $\rightarrow$ AI?

Evolution shows how complexity and solutions to the problem of survival arise from populations, selection and recombination.

In our search for AI, such ideas are appealing:

- intelligence as an emergent property
- optimisation of behaviours
- adaptation to changes
- learning of new behaviours
- searching for optimal solutions (or better than the rest?)
- Artificial Life

Using These Ideas

How does all this inspire us to build clever computer programs?

- find a hard problem (local maxima)
- encode solutions as a genotype
- map genotypes to phenotypes (optional)
- evaluate phenotype for fitness
- mate fit genotypes using crossover
- make a few mutations
- continue until you have your solution (or you are convinced that no improvement is possible)

The Main Issues

- How do I represent a solution?
- How should I rate a solution for fitness?
- How large should the population of solutions be?
- How much selection pressure should I apply?
- What form of crossover should I use?
- what form of mutation should I use?

Overview

Towards a Canonical GA

- Darrell Whitley (1989) The GENITOR algorithm and selective pressure. Why rank-based allocation of reproductive trials is best.
  [GENetic ImplemeTOR]
- A heuristic fitness function is not a measure of any "exact fitness"
- Ranking introduces a uniform scaling across the population
- Direct control of selective pressure
- Efficient coverage of the search space

see: D. Whitley: A genetic algorithm tutorial. Statistics and Computing (1994) 4, 65-85

Example: Evolution of a walking machine

- Parameters: Number and type of joints, length of links, mass distributions, shape of feet, position and strength of springs, actuators, …
- Binary encoding of all parameters
- Choice of a fitness function:
  1. fitness 1 for a reasonable gait, otherwise 0
  2. number of steps (from a defined initial position)
  3. distance walked
  4. Some combination of the above plus:
     speed, efficiency, symmetry, similarity to natural legs, to natural gaits, …

Conventions

- old population
- selection
- intermediate population
- recombination mutation
- new population

} one generation

- An *individual* is a string (genotype, chromosome)
- Fitness values are replaced by ranks (high to low)
- Fitness = objective function = evaluation function

Roulette Wheel Selection

(I. Plain variant)

mean fitness $\bar{f} = \frac{1}{n} \Sigma_i f_i$

normalized fitness: $f_i / \bar{f}$

(from now on short: fitness)

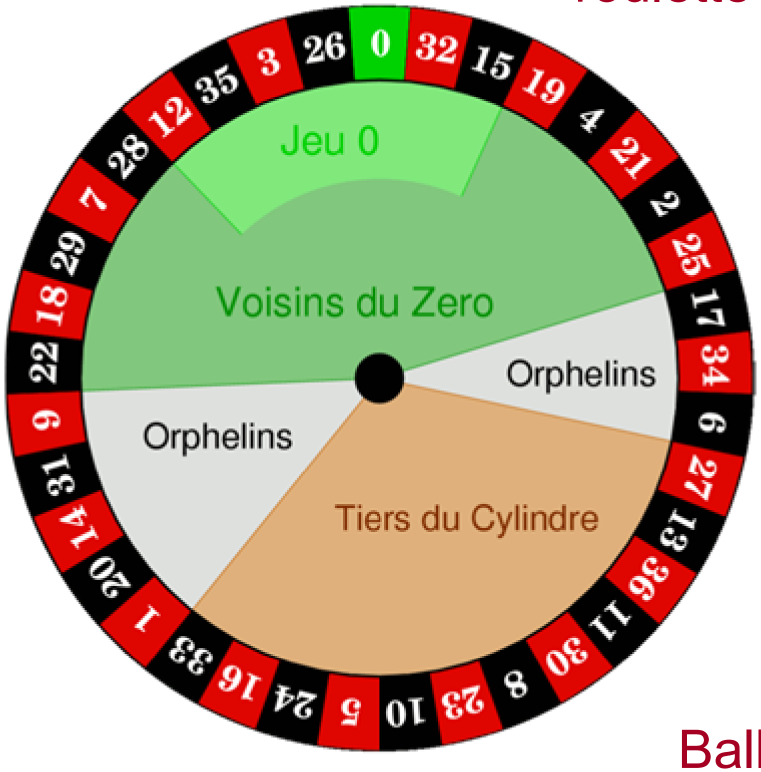
Intermediate population:
Ratio of fitness to average fitness determines number of offspring:

Stochastic sampling with replacement

A new individual is a copy of an old individual (of fitness f_i) with probability $f_i / n\bar{f}$

If $f_i = \bar{f}$: the individual "survives" with probability $1-(1-1/n)^n$

Sector (French) bets in roulette



Ball spins n times

Roulette Wheel Selection

(II. A more intuitive & practical variant)

mean fitness $\overline{f} = \frac{1}{n} \Sigma_i f_i$

normalized fitness: $f_i / \overline{f}$

(from now on short: fitness)

Both variants are equivalent in the sense that they produce an unbiased sample of the fitness in the population

Now: Only the outer wheel with equidistant pointers spins once and pointers in each sector are counted

Remainder stochastic sampling

If $f_i = \overline{f}$: the individual "survives"

If $f_i / \overline{f} < 1$: probability of "survival"

If $f_i / \overline{f} > 1$: number of offspring int($f_i / \overline{f}$)

(plus a probability accounting for the non-integer part)

From intermediate to new population

Preparation:

- Population was already shuffled by selection
- Individuals are strings of equal length *L*
- Choose a probability p_c

Crossover:

- Choose a pair of individuals
- With probability p_c:
  - choose a position from 1 to *L*-1
  - cut both individuals after this position
  - re-attach crossed: xyxxxyyy, abbabbab → xyxxbbab, abbaxyyy
- Move the obtained pair to the new population
- Repeat for the remaining pairs (assume n even)

From intermediate to new population

Preparation:

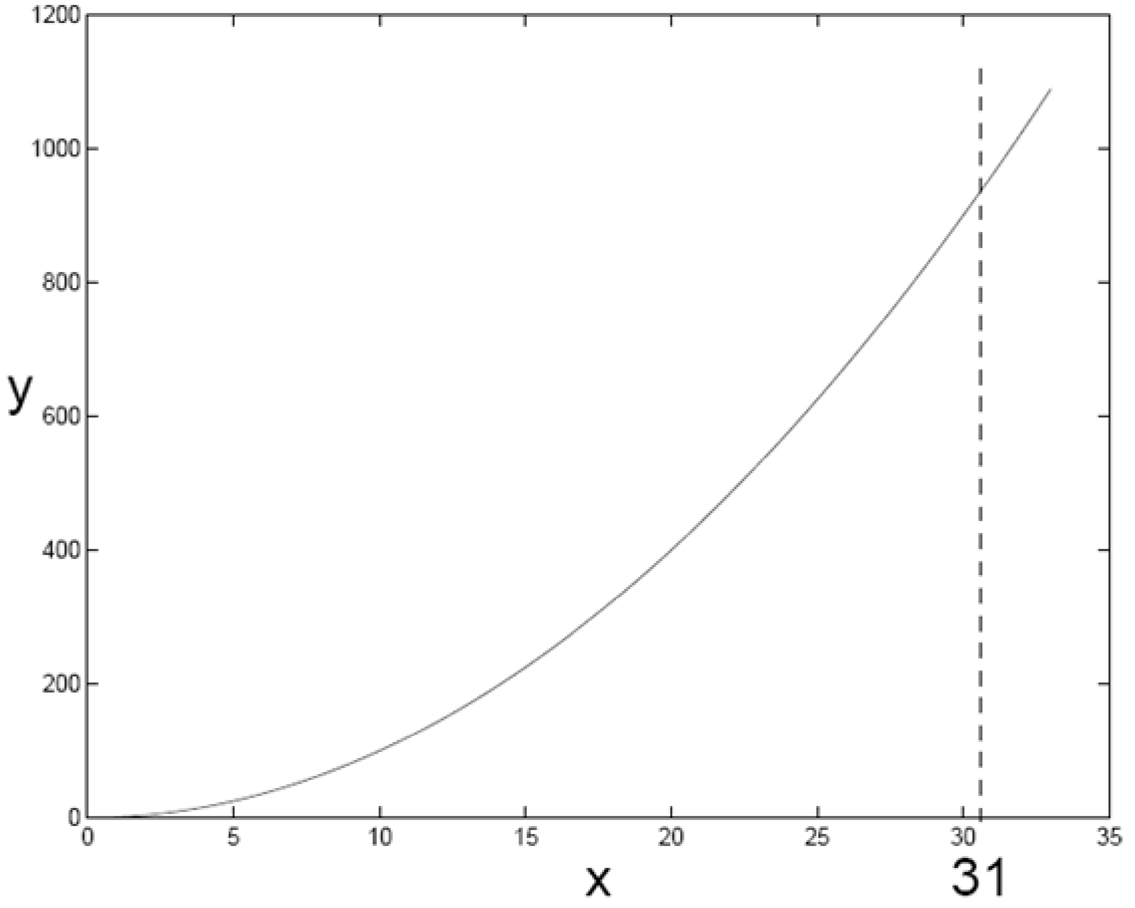
- Crossover finished
- Individuals are strings of length *L* made from *k* different char's
- Choose a probability $p_{m\ (small)}$ (possibly rank-dependent)

Mutation:

- Choose an individual
  - for each position from 1 to *L*
  - with probability p_m:
  - set the character (bit if binary) to a random value
  - or change it [this gives k/(k-1) (twice if binary) the effect!]
- Move the obtained mutant to the new population
- Repeat for the remaining individuals

A Simple Example

Maximise $y = x^2$ for x in the range 0 to 31. (What is the answer?)



A Simple Example

Represent x as 5 bits:

00000	0	0
00001	1	1
00010	2	4
⋮		
11111	31	961

Here raw fitness values will be used instead of ranks. Does this make any difference?

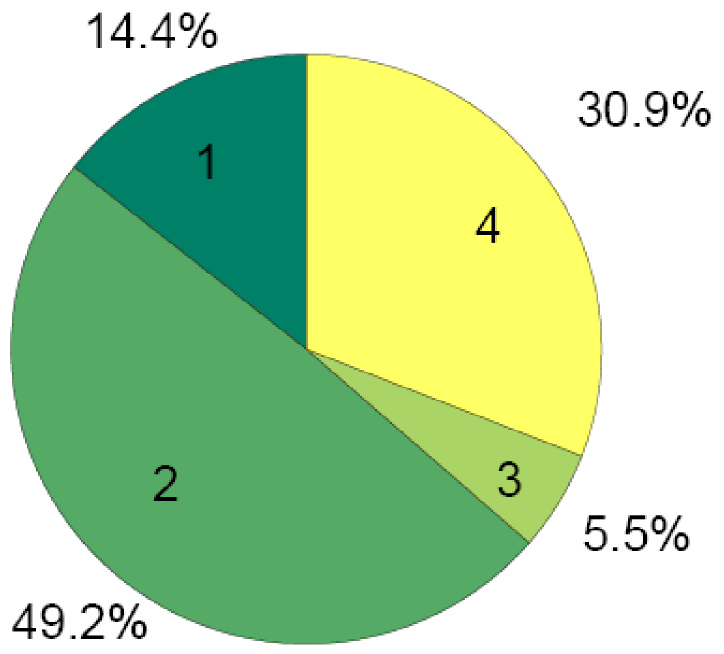
Use a population of size 4 (far too small!)

Initial population:

i	Value	Evaluation	% of Total (= prob. of being selected)
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9

A Simple Example

Selection:



So our intermediate population might be: 1, 2, 2, 4

A Simple Example

Create next generation:

$p_c = 1.0,\ p_r = 1 - p_c = 0.0$ – so crossover is always applied here

Pair parents randomly, choose crossover points randomly:

Parents: 1 and 2

$$\begin{array}{lcl} 0110\ 1 & & 0110\ 0 \\ & \longrightarrow & \\ 1100\ 0 & & 1100\ 1 \end{array}$$

Parents: 2 and 4

$$\begin{array}{lcl} 11\ 000 & & 11\ 011 \\ & \longrightarrow & \\ 10\ 011 & & 10\ 000 \end{array}$$

A Simple Example

Mutation: $p_m = 0.001$, 20 bits. No mutation here $(20 * 0.001 = 0.02)$.

New population:

i	Value	Evaluation	% of Total
1	01100	144	8.21
2	11001	625	35.63
3	11011	729	41.6
4	10000	256	14.6

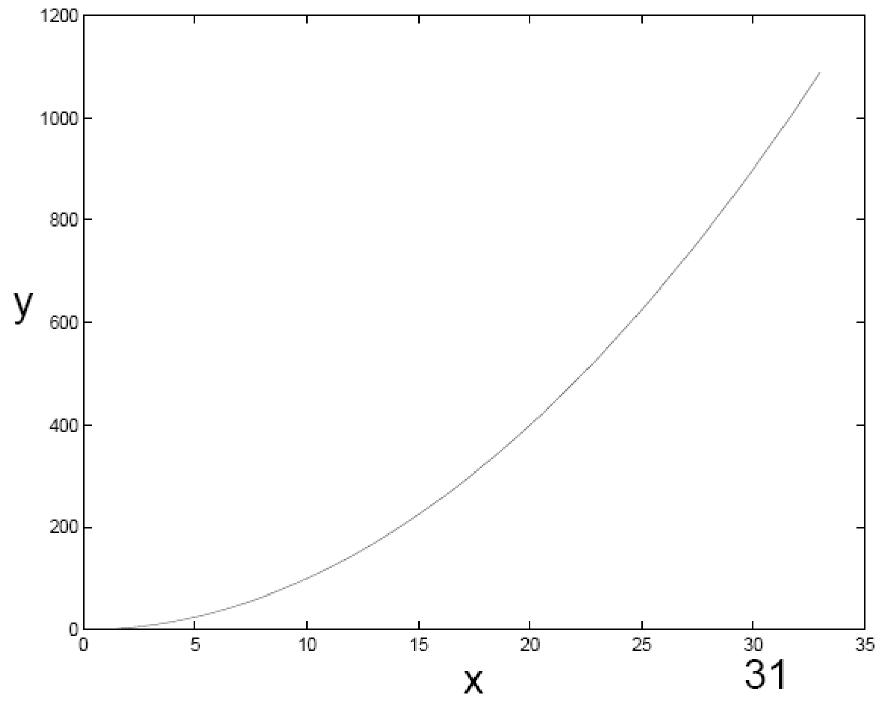
What is the average evaluation of this population?

How does it compare with the average evaluation of the previous generation?

Continue until no improvement in the best solution for k generations, or run for fixed number of generations.

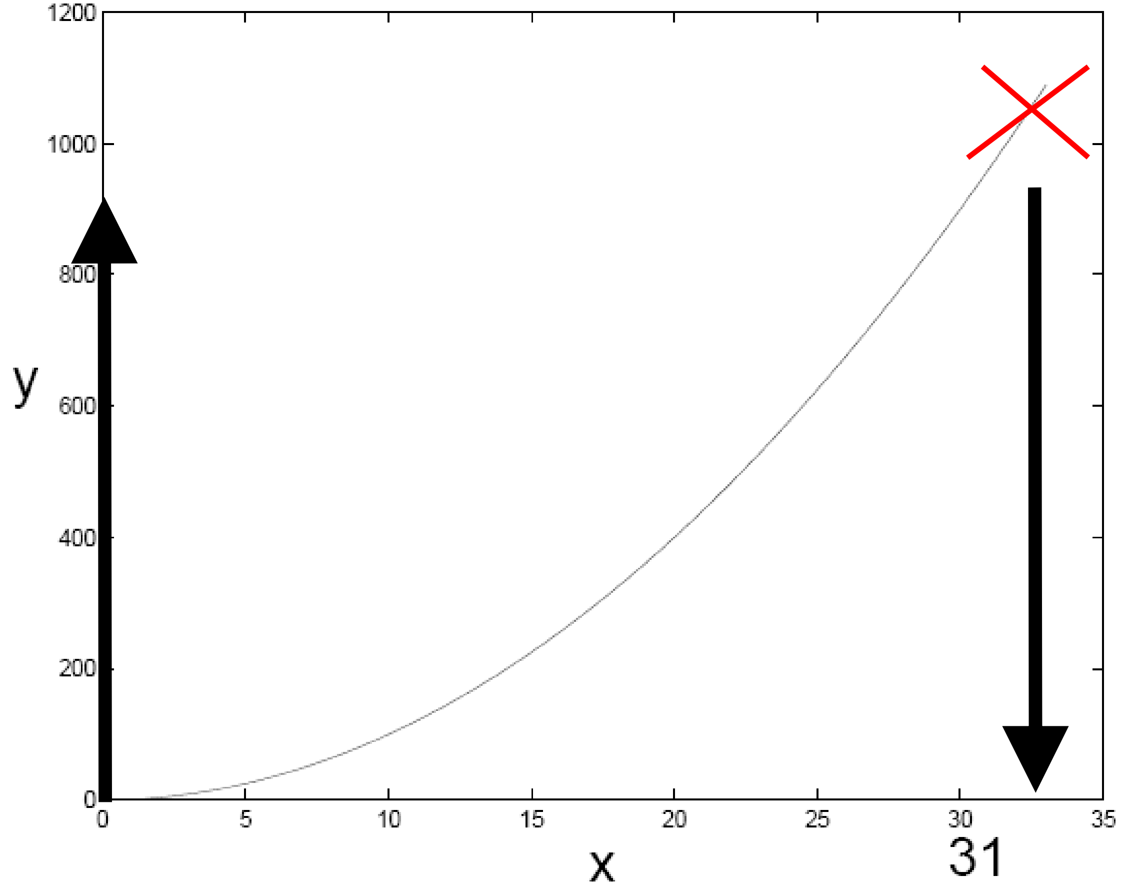
A Simple Example

How does it work? What is the best solution?



Climbing up the fitness curve, putting together building blocks of good sub-solutions.

A “deceptive” fitness function



$$f(x) = \begin{cases} 961 \text{ for } x=0 \\ x^2 \text{ for } 0<x<31 \\ 0 \text{ for } x=31 \end{cases}$$

?

Next time: More examples. Search spaces. Schemas.