Genetic Algorithms and Genetic Programming

Lecture **4**: (6/10/09)

The schema theorem and building block hypothesis

Michael Herrmann

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

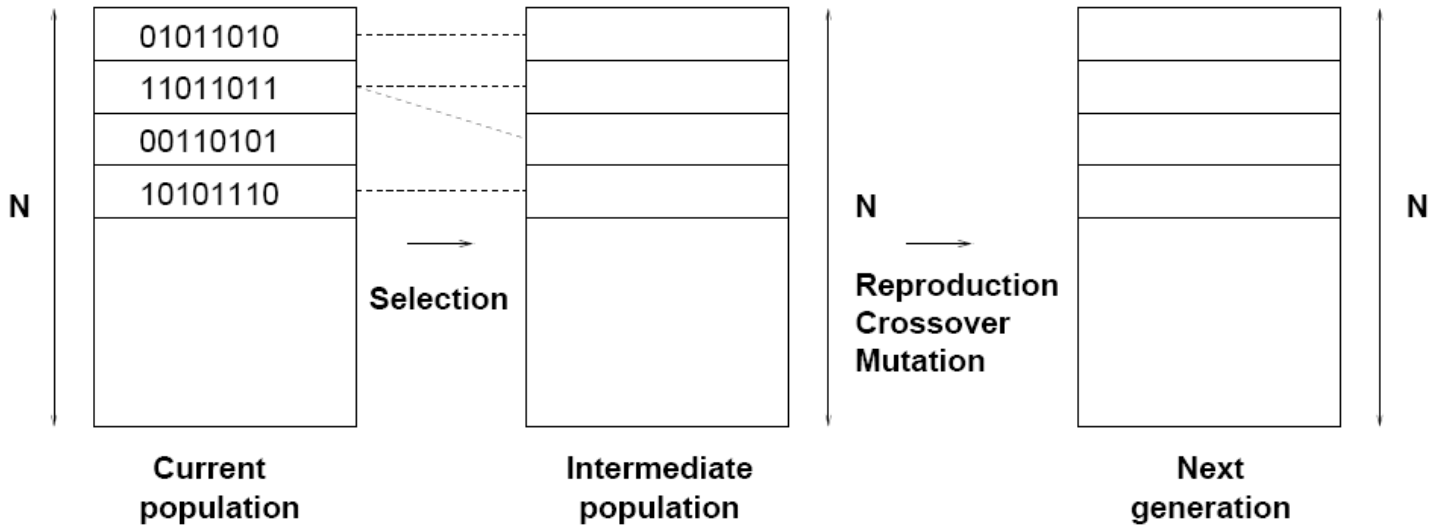
Overview

1. Introduction: History
2. The genetic code
3. The canonical genetic algorithm
4. Examples & Variants of GA
5. The schema theorem

The Canonical GA: Short Overview

Repeat
- Evaluate fitness
- Select intermediate population
- Do crossover or reproduction
- Do mutation

Until solutions good enough



The Canonical GA: Overview

Evaluation function F gives a score f_i to each individual solution i.

If $\bar{f}$ is the average evaluation over the whole population of N individuals, then the **fitness** of i is $f_i/\bar{f}$

Probability of selection of solution with evaluation f_i is $f_i/\Sigma_i f_i$

Roulette Wheel selection

This is the step that most people get wrong in exams:

Select two parents **at random** from the intermediate population. Apply **crossover** with probability p_c, with probability $1 - p_c$ copy the parents unchanged into the next generation – **reproduction**.

Crossover: from the 2 parents create 2 children using 1-point, 2-point, n-point crossover. Select crossover point **uniform-randomly**:

Mutation: take **each bit** in turn and with Prob(mutation) $= p_m$, flip it ($0 \rightarrow 1$, $1 \rightarrow 0$). $p_m < 0.01$ usually. Note that the probability p_m is applied differently from p_c.

This is one **generation**. Do for many generations, till solutions are optimal or good enough.

The “Philosophy” of GA

- Encoding: Create a space of solutions
- Fitness function: Discriminate good from bad solutions
- Initialization: Start with good candidate solutions
- Selection: Prefer better solutions to worse ones
- Recombination: Combine parental traits in a novel manner
- Mutation: Creating individual traits by random local search
- Termination: Comparing achieved and achievable fitness

How do the simple mechanisms create something useful when combined?

- Selection + Mutation = Continual improvement
- Selection + Recombination = Innovation

From David E. Goldberg: The design of innovation: lessons from and for competent genetic algorithms.

Variants of GAs

- Selection:
  - Roulette wheel (see last lecture)
  - Tournament selection (select a pair and keep two copies of the better one)
  - Elitism (best individuals are moved unchanged to the next generation)
  - Insertion of a few new random individuals in each generation
- Crossover:
  - 1-point, 2-point, …, *n*-point
  - cut and splice (a different cutting point in each of the parents, children of different length)
  - half-uniform crossover scheme (exactly half of the nonmatching bits are swapped)
  - more than two parents; islands (crossover mostly within groups)
- Mutation:
  - point mutation: flip or random
  - exchange two randomly chosen characters (perhaps coupled mutations)
  - inversion
  - fitness-dependent, adaptive mutation rates etc.

Termination of a GA

The generational process is repeated until a termination condition has been reached, e.g.

- A solution is found that has optimal fitness
- Fitness indicates a sufficient improvement over alternative algorithms
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The fitness of the highest ranking solution is reaching or has reached a plateau such that successive iterations no longer produce better results
- The diversity of the population has vanished
- Combinations of the above
- Decide: really finish or restart a variant of the GA on the same task

must read: http://en.wikipedia.org/wiki/Genetic_algorithm

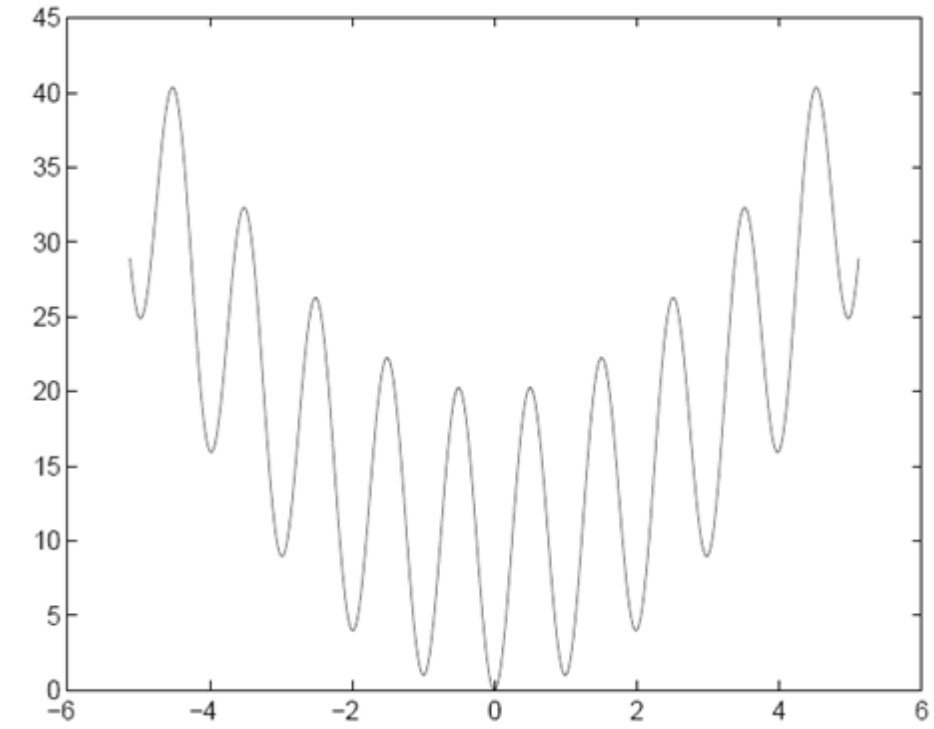
Tournament selection vs. Roulette Wheel selection

- Roulette Wheel selection (see last lecture)
  - may be used on (raw) fitness values or rank
  - chance of survival in a single run (for rank *i*): $p=(2i)/(n^2+n)$ (at least one from *n* runs $P=1-(1-p)^n$ for the first variant)
  - best (rank *n*): $p=2/(n+1)$, worst (rank *1*): $p=2/(n^2+n)$
  - roulette wheel with elitism is fairly similar to tournament
- Tournament selection (*n* winners from *n* tournaments)
  - chance of survival depends on rank [selection for tournament may also depend on rank]
  - $P=(i-1)/(n-1)$
  - best (rank *n*) individual beats any other: $P=1$
  - worst (rank *1*) $P=0$
  - outcome of a tournament may be stochastic (add elitism)
  - main advantage: Can be used if fitness function cannot be calculated explicitly, e.g. in the evolution of chess players
  - better parallelizable

Example: Function Optimization

Minimise Rastrigin's Function:

$$f(x) = 10 + x^2 - 10\cos(2\pi x),\ -5.12 \leq x \leq 5.12$$



Representation: binary strings

$$x = x_{min} + b(x_{max} - x_{min})/(2^m - 1)$$

So for 8-bit strings

$$x = -5.12 + b(5.12 - -5.12)/(2^8 - 1)$$

If $b = 10011001$ then this represents

the integer 153, so

$$x = -5.12 + (153 \times 10.24/255) = 1.024$$

Solution: $x = 0.0201\ f(x) = 0.0799$ rather than: $x = 0, f(x) = 0$ Why?

More on this example:

search for "Rastrigin" at www.mathworks.com, www.obitko.com/tutorials/genetic-algorithms/

Overview

1. Introduction: History
2. The genetic code
3. The canonical genetic algorithm
4. Examples & Variants of GA
5. The schema theorem

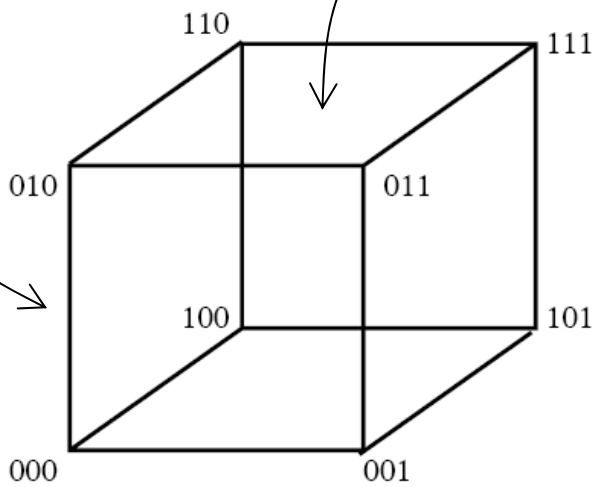
Search spaces as Hypercubes

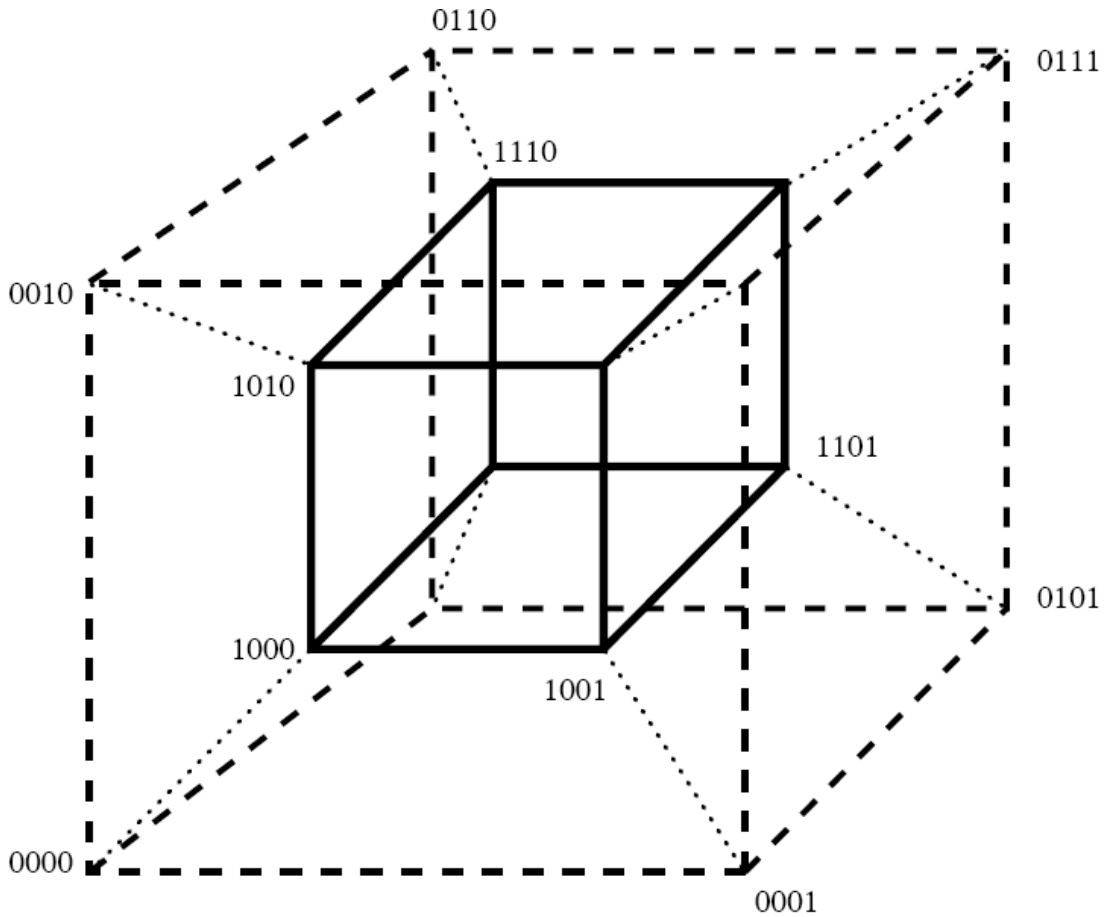
Binary encoding: solution "*c*" in $\{0,1\}^L$

e.g. *c=(0,1,0)* for *L=3* or *c=(0110)* for *L=4*

(0,,0) denotes a line*

(,1,*) denotes a plane*





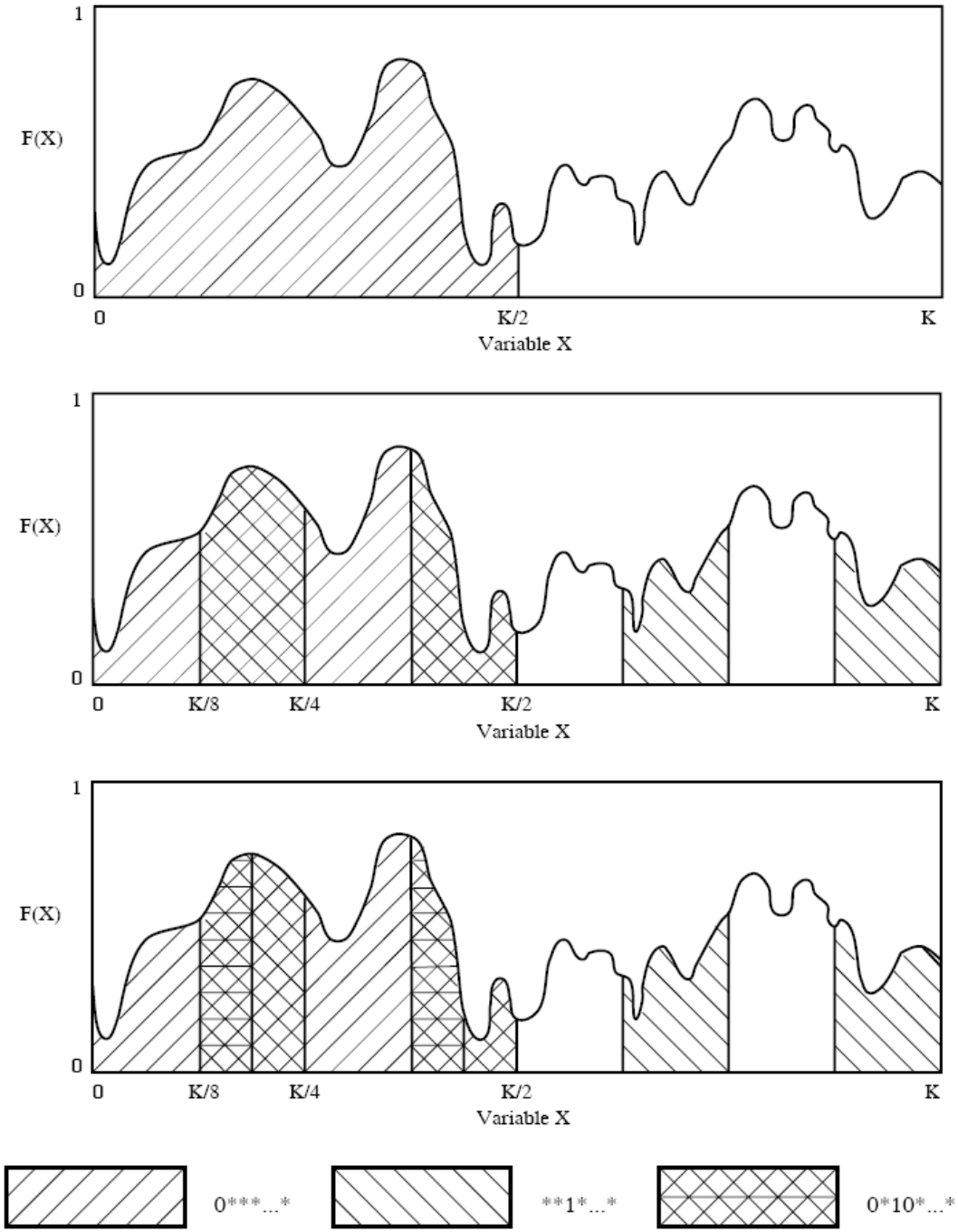
Schemata (J. Holland, 1975)

- All (inheritable) features of the phenotype are encoded by schemata
- A schema is a string that contains wildcards (“*”)
- The order of the schema is the number of bits that are actually there
- E.g. **01***1 is a schema of order 3 (and length 8)
- each chromosome is a corner of the hypercube
- There are 3^L-1 different schemata
  (not counting the schema of order 0: ** … *)
- each chromosome is part of 2^L-1 hyperplanes
- Implicit parallelism: Each individual samples many hyperplanes

Binary encoding of a 1-D variable

Fitness of a schema is the average over the corresponding hyperplane (or rather the sample across the population)

Sampling of the hyperplanes is essentially unaffected by local optima



From: Whitley, 1992

How Do GAs Work?

The Schema Theorem

$$E(m(H,t+1)) \geq \frac{\hat{u}(H,t)}{\bar{f}(t)} m(H,t)(1 - p_c \frac{d(H)}{L-1})[(1-p_m)^{o(H)}]$$

$f(s_i,t)$ fitness at time t of solution s_i

$m(s_i,t)$ is the number of copies of s_i in the population at time *t*

$\bar{f}(t)$ is the average fitness of the population at time *t*

$$E(m(s_i,t+1)) = m(s_i,t)\frac{f(s_i,t)}{\Sigma_j f(s_j,t)} P$$

P denotes the population size

$E(\cdot)$ is the expected value $\frac{f(s_i,t)}{\Sigma_j f(s_j,t)}$ is the probability of selecting s_i

Writing $\Sigma_j f(s_j, t)/P$ as $\bar{f}(t)$

$$E(m(s_i, t+1)) = m(s_i, t)\frac{f(s_i, t)}{\bar{f}(t)}$$

proportion of the population that is s_i

$$\text{Prop}(s_i, t+1) = \frac{m(s_i, t)}{|P}\frac{f(s_i, t)}{\bar{f}(t)}$$

So above-average-fitness strings get more copies in the next generation and below-average-fitness strings get fewer.
Suppose s_i has, and continues to have, an above-average-fitness of $(1+c)\bar{f}$.
Then for c>0

$$E(m(s_i, t+1)) = m(s_i, t)\frac{\bar{f} + c\bar{f}}{\bar{f}} = (1+c)\, m(s_i, t)$$

If we have $m(s_i, 0)$ copies at $t = 0$, then we have $m(s_i, t) = (1+c)^t\, m(s_i, 0)$
–this gives **exponential growth** – and decay for $c < 0$.
–So fit solutions come to dominate

Fitness of Schemata

If solutions s_i, s_j, s_k all sample the same schema H, we can calculate the **average fitness** $\hat{u}$ **of** H from the fitnesses of the m solutions that sample it:

$$\hat{u}(H,t) = \frac{\sum f(s_i), f(s_j), f(s_k), \ldots}{m(H,t)}$$

Given $m(H,t)$ and $\hat{u}(H,t)$, can we calculate $m(H,t+1)$?

$\hat{u}(H,t)$ is the average fitness of H at time t
$m(H,t)$ is the number of instances of H at time t
$\bar{f}(t)$ is the average fitness of the population at time t

How many instances of H will be present in P after selection?

Proportion:

$$\text{Prop}(H) = \frac{m(H,t)}{P}\frac{\hat{u}(H,t)}{\bar{f}(t)}$$

first component of **the Schema Theorem**

after P spins : $$E(m(H,t+1)) = m(H,t)\frac{\hat{u}(H,t)}{\bar{f}(t)}$$

other parts of the Schema Theorem:

Defining length is the distance $d(H)$ between the first and last bits of the schema

1 1 * * 1 1 0 * defining length 6
* * * * 1 1 0 * defining length 2

l : total string length

$$\Pr(\text{surviving crossover}) = 1 - p_c\frac{d(H)}{l-1}$$

$$\Pr(\text{surviving mutation}) = (1-p_m)^{o(H)}$$

$$E(m(H,t+1)) = m(H,t)\frac{\hat{u}(H,t)}{\bar{f}(t)}(1 - p_c\frac{d(H)}{l-1})[(1-p_m)^{o(H)}] \quad (?)$$

The Schema Theorem

schemata can be **created** through crossover and mutation. So we need a $\geq$.

$$E(m(H,t+1)) \geq m(H,t)\frac{\hat{u}(H,t)}{\bar{f}(t)}(1-p_c\frac{d(H)}{l-1})[(1-p_m)^{o(H)}]$$

Highest when

- $\hat{u}(H,t)$ is large – fit
- $d(H)$ is small – short
- $o(H)$ is small – small number of defined bits

Schema Theorem in words: short, low-order, above average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm.

Outlook

- More examples
- Implications of the schema theorem
- Criticism of the schema theorem
- Hybrid algorithms