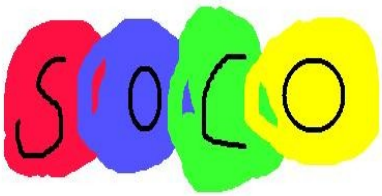
An introduction to Genetic Algorithms

Lluís A. Belanche
Computational Intelligence
2015-16



Soft Computing Research Group

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Outline of various techniques

- Genetic Algorithms (bitstrings)
- Evolutionary Programming (finite-state automata)
- Evolution Strategies (real-valued vectors)
- Genetic Programming (computer programs)
- Classifier Systems (rules)

Genetic Algorithms

Genetic Algorithms are adaptive search algorithms for which:

- Individuals are bit strings
- Strategy is (μ , λ) with $\mu = \lambda$
- Mutation is seen as transcription error
- Recombination is called crossover
- Selection is proportional to fitness

Genetic Algorithms

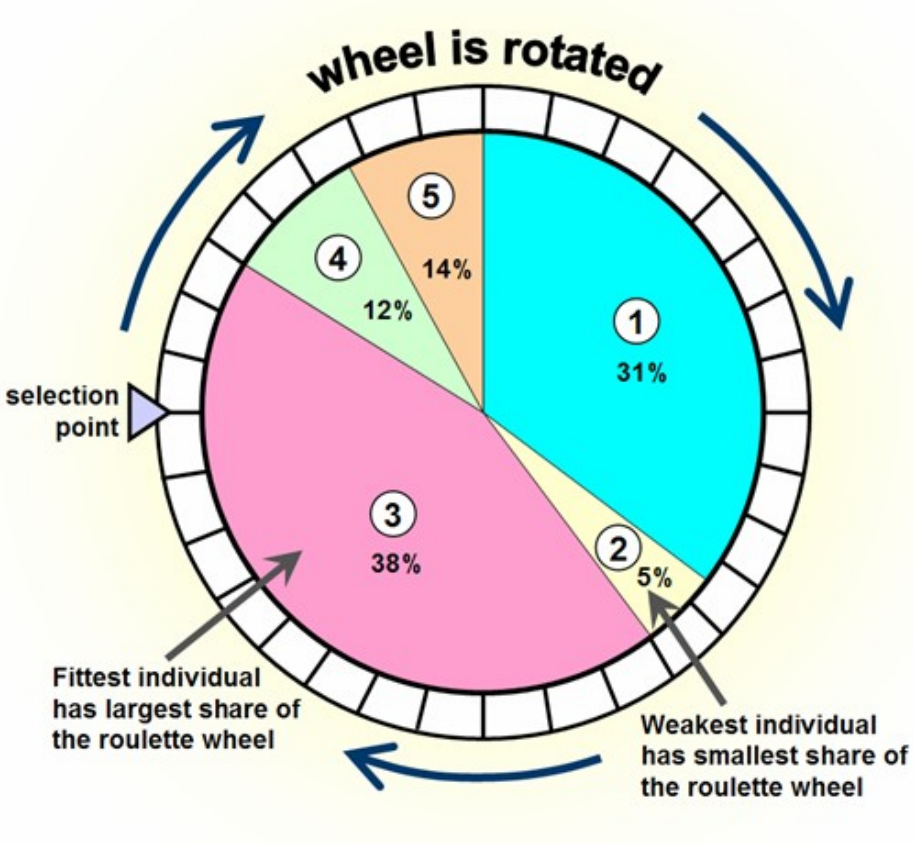
- In solving a given optimization problem with a GA, the first task would be **describing the solution domain efficiently** and **devising an appropiate description of a solution** in terms of a bitstring:
  - How many loci? (chromosome length)
  - Do we allow polyploidy? (dominant/recessive traits)
  - What exactly are the genes? Do they come in groups?
  - Using a bijective code (at least, injective)
  - Devising correct recombination & mutation operators (sound, complete, scalable)

Selection – first (good?) ideas

- Selection completely at random
  → no benefit for being better than others
- Always select (only) the best
  → very high selective pressure and little exploration
- **Stochastic selection** (better fitness = higher chance of reproduction)
  → leads to many variations

Selection – The Roulette Wheel

- Each solution gets a region on a roulette wheel according to its fitness
- Spun wheel, select solution marked by roulette-wheel pointer
- **stochastic selection** (better fitness = higher chance of reproduction)



http://www.edc.ncl.ac.uk/highlight/rhjanuary2007g02.php

Selection – alternatives to Roulette Wheel

Standard Roulette Wheel may be very sensitive to the "details" of the fitness function

- Spun wheel once with as many equally-spaced pointers as individuals (*stochastic universal sampling*), or
- Give a sure number of copies and perform standard Roulette Wheel with remainder (*remainder stochastic sampling*), or use
- *Rank selection* (probability of selection is proportional to rank), which can be used to cope with:
  - High selective pressure in the first few generations
  - High selective pressure due to a decrease in fitness variance as the search proceeds

Selection - Elitism

- The best A individuals from the last B generations are *kept unchanged* for the next generation

Example:

- keep the best individual of current population (A=1,B=0)
- unrealistic but ensures best fitness of a generation never decreases
- entails a decrease in diversity (EXPLOITATION)
- they can be subject to specific local improvement
  - Hill-climbing
  - Large mutation steps

Selection - Tournament

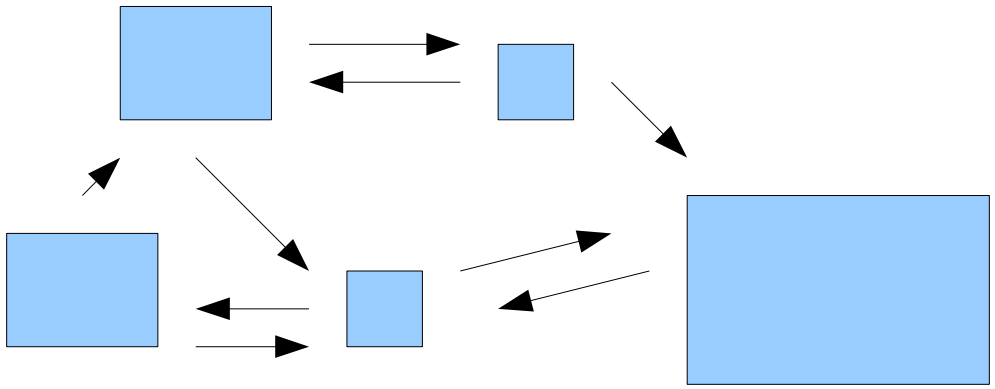
- Randomly select *k* individuals (with replacement)
- **Winner**: individual with best fitness among these *k* individuals
- **Example**:
  - randomly pick two individuals and keep the best of the two; do this $\mu-1$ times (these parents will then undergo recombination)
  - use **elitism** (once) to restore population size μ

Selection - Niching

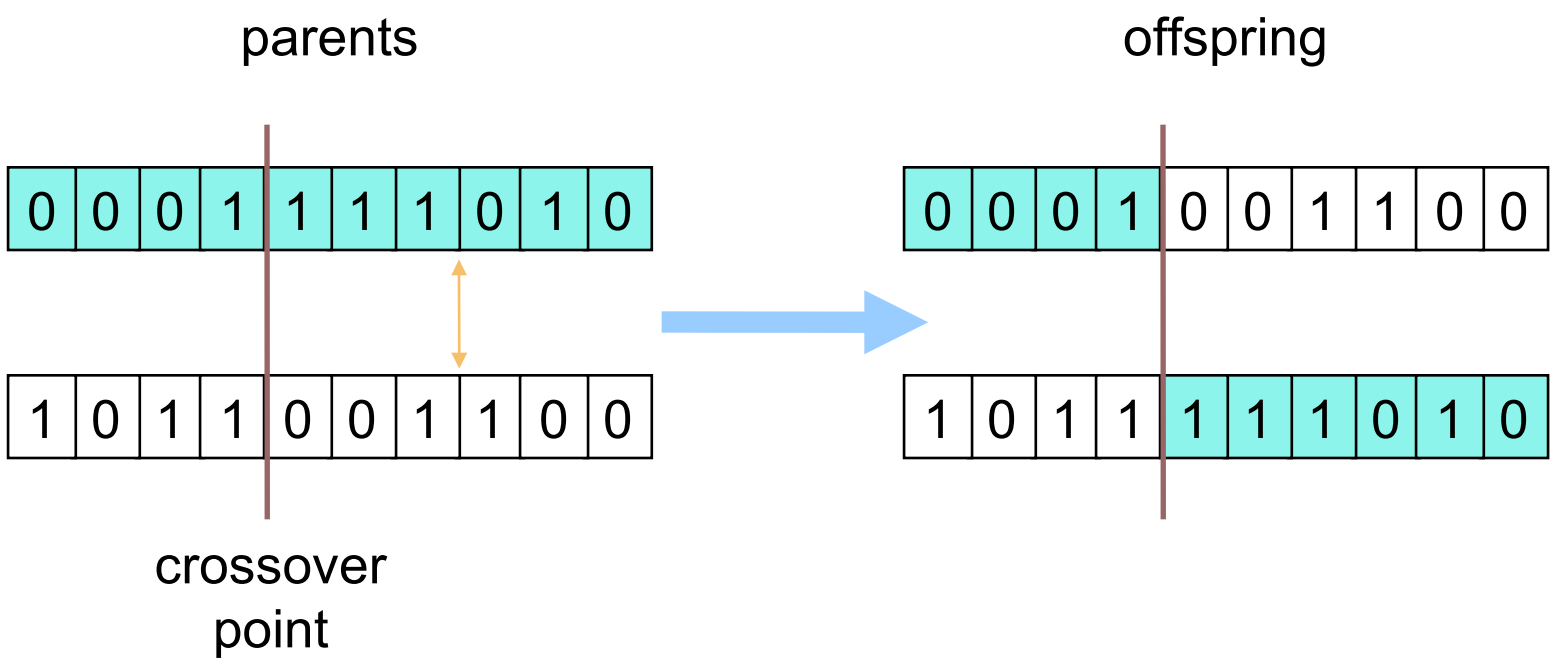
Start with a single population

Split it into parts (niches) around promising parts of the search space

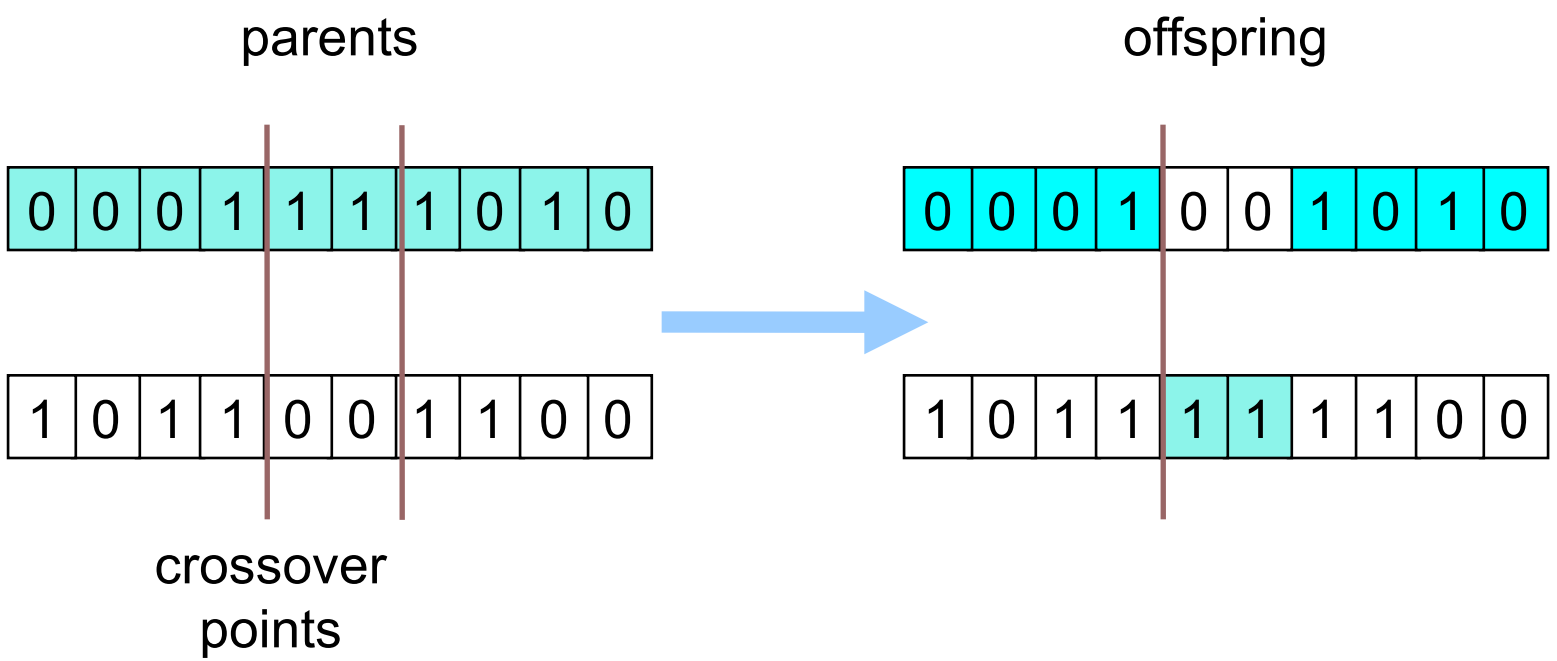
Eventually let them interchange information



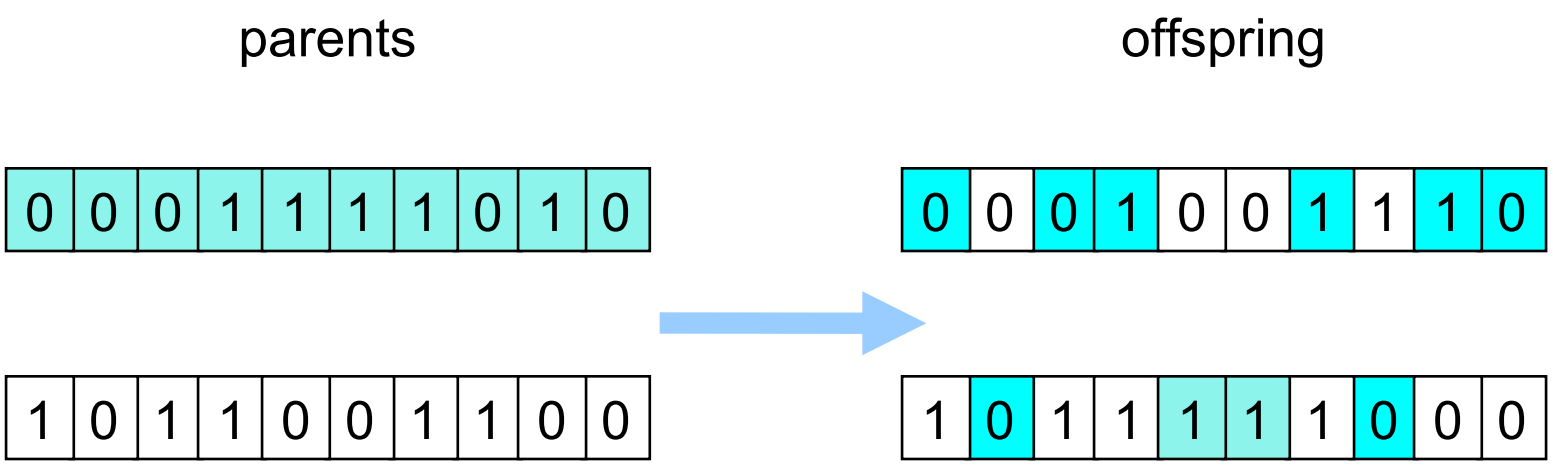
Crossover: One Point Crossover (1P)



Crossover: Two Point Crossover (2P)



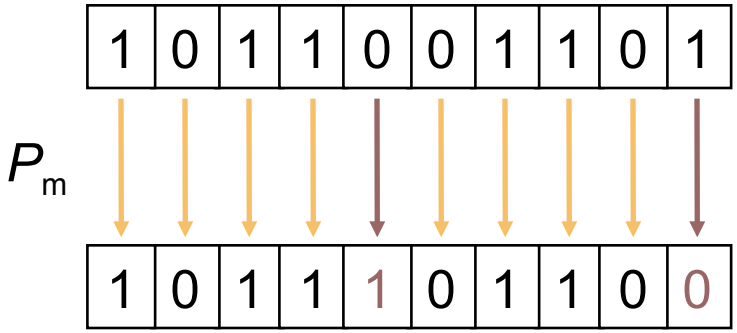
Crossover: Uniform Crossover (UX)



UX evaluates each bit in the parent strings for exchange with a probability of 0.5

HUX (Half Uniform Crossover) is as UX, but exactly half of the nonmatching bits are swapped

Mutation



independent Bernoulli transcription errors

Mutations occur with some small probability every time a chromosome is duplicated

In humans, this probability is around 10^{-8} (the chance that a given nucleotide is mutated in a generation)

Knowing when to stop

- Limit the number of generations, or of fitness evaluations
- Detect convergence:
  - No relative improvement in the fitness evaluation
  - No change in best
  - Uniformity of population or fitness evaluations
  - Fitness evaluation beyond a predefined value

Replacement strategies

- **Generational** genetic algortihms (GGA):
  - replace all parents with their offspring
  - no overlap between populations of different generations
- **Steady-state** genetic algorithms (SSGA):
  - immediately after offspring is created and mutated, it is used to replace some parents in the old generation
  - some overlap exists between populations of different generations

Replacement strategies

- Replacement strategies for SSGA:
  - the offspring replaces the **worst** individual of the current population
  - the offspring replaces a **random** individual of the current population
  - the individual to be replaced is selected using **tournament selection** (now for the worst one)
  - the offspring replaces the **oldest** individual of the current population

Schools of “thought”

- In GAs, originally there were two different views of what a population represents:
  - The **Michigan** view: individuals represent parts of the solution; the whole population represents a full solution
  - The **Pittsburgh** view: each individual represents a full solution
- Example: evolving neural networks

Binary Representations

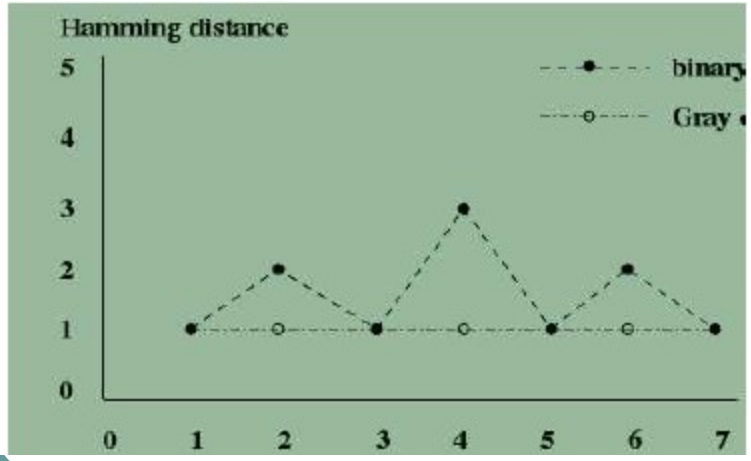
The (variables in the) optimization problem may:

- admit a natural binary representation, or
- be given by nominal-valued variables
- be given by a discrete structure (tree, graph, ...)
- be given by continuous information (real numbers)

Binary Representations

Problems using a binary representation:

- maximum attainable precision may be less than needed to represent the optimum
- introduction of Hamming cliffs



Hamming cliff is formed when two numerically adjacent values have bit representations that are far apart

$7_{10} = 0111_2$ vs $8_{10} = 1000_2$

Large change in solution is needed for small change in fitness

Gray Coding

In a Gray coding, the Hamming distance between the *representation* of consecutive numerical values is 1

Example for l=3:

	Binary	*Gray*
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Converting binary strings to Gray bit strings:

$$g_1 = b_1$$
$$g_l = b_{l-1}\overline{b}_l + \overline{b}_{l-1}b_l$$

where b_l is bit l of the binary number

$$b_1 b_2 \cdots b_{n_b}$$

How a GA might work: schemata theory

A **schema** is a template describing a subset of individuals:

*	*	*	1	0	*	1	*	*	*

Don't care symbol: *

order of a schema: $o(S)$ = # fixed positions

defining length $\delta(S)$ = distance between first and last fixed position

there are 3^n possible schemata

there are (δ over o) 2^o schemata of order o and defining length δ

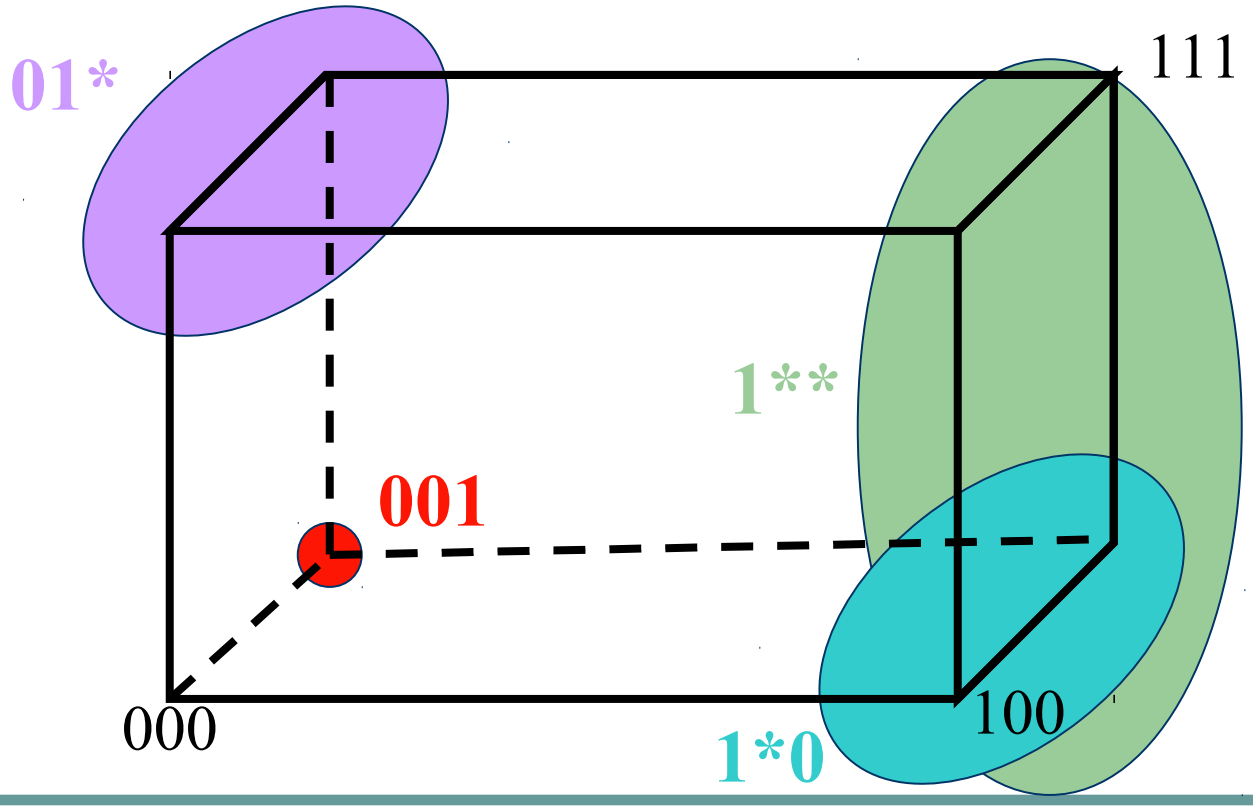
a schema S includes (represents) $2^{l - o(S)}$ individuals

an individual of length l belongs to (i.e. represents) 2^l schemata

(this one does not depend on the alphabet being binary)

Schemata as hyperplanes

A schema (*pl.* schemata) is a string in a ternary alphabet (0,1,*) representing a hyperplane in the search space.



Implicit Parallelism

In a population of μ individuals of length l:

$$2^l \leq \text{ number of processed schemata } \leq \mu \cdot 2^l$$

$\rightarrow$ far more schemata than individuals are *implicitly* processed (not disrupted by crossover and mutation) [Holland 1989]:

Lower bound of the order of $\mu^3/\sqrt{(\log_2 \mu)}$

--see Bertoni & Dorigo (1993)

"Implicit Parallelism in Genetic Algorithms"

Artificial Intelligence **61**(2), p. 307–314

Schemata interpretations

- **Geometric** view: hyperplanes in $\{0,1\}^l$
- **Evolutionary** view: what should survive is not the individuals, but their genetic stuff ($\rightarrow$ the schemata)

Thus the GA is a **schema processor**

The Schema Theorem

Short, **low-order**, **above-average** individuals (and, implicitly, the schemata that they represent) receive **exponentially increasing** trials in subsequent generations.

Schema theory is (generally) accepted to be correct, providing intuitive explanation for the good (and bad) performance observed in practical GA applications

Markov chains are an alternative formalism

The Building Blocks Hypothesis

Closely related to the Schema Theorem is the “Building Block Hypothesis” [Goldberg 1989]:

“Genetic Algorithms work by discovering and exploiting **building blocks** --groups of closely interacting genes-- and then successively combining these (via crossover) to produce successively larger building blocks until the problem is solved.”

Deception

A problem is said to be **deceptive** if the building blocks identified actually lead the GA away from the global objective.

f(000) = 28	f(001) = 26
f(010) = 22	f(100) = 14
f(110) = 0	f(011) = 0
f(101) = 0	f(111) = 30

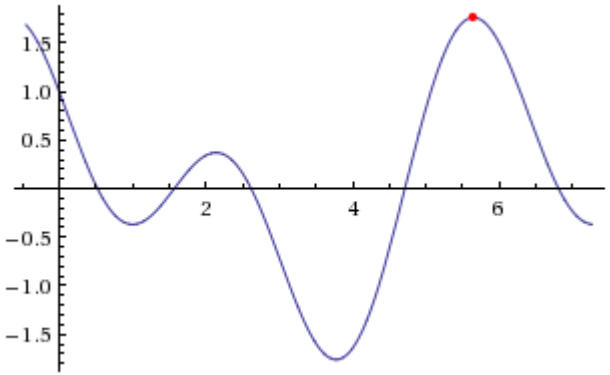
F(0**) > f(1**)	F(00*) > f(11*), f(01*), f(10*)
F(*0*) > f(*1*)	F(0*0) > f(1*1), f(0*1), f(1*0)
F(**0) > f(**1)	F(*00) > f(*11), f(*01), f(*10)

(global optimum is 111; 000 is suboptimal)

Example: Function optimisation (1)

Let $f(x) = \cos(x) - \sin(2x)$

(to be maximised in $[0, 2\pi]$)

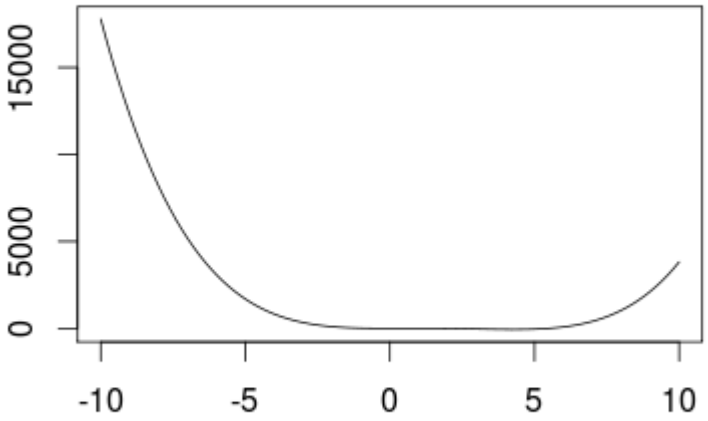


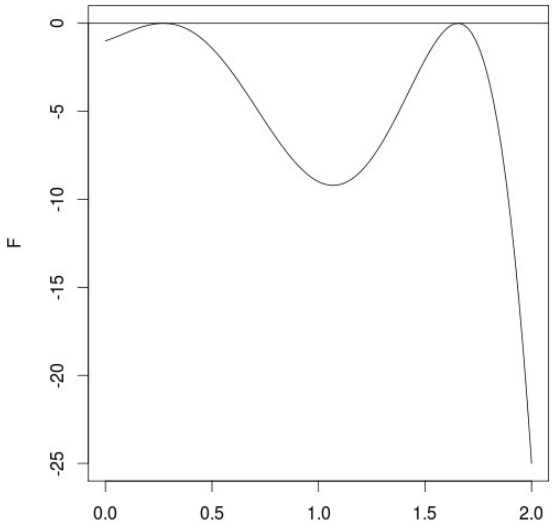
Example: Function optimisation (2)

Finding the roots of a polynomial p(x) in an interval

Fitness function: maximize $F(x) = -p(x) \cdot p(x)$

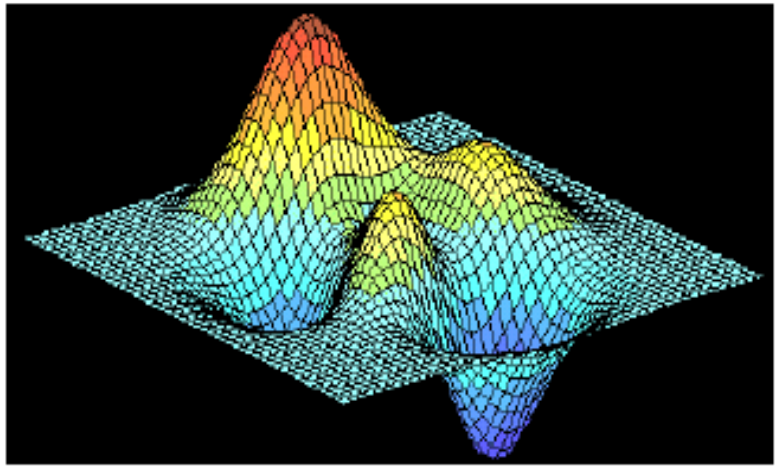
Example: $p(x) = x^4-7x^3+8x^2+2x-1$ in [0,2]

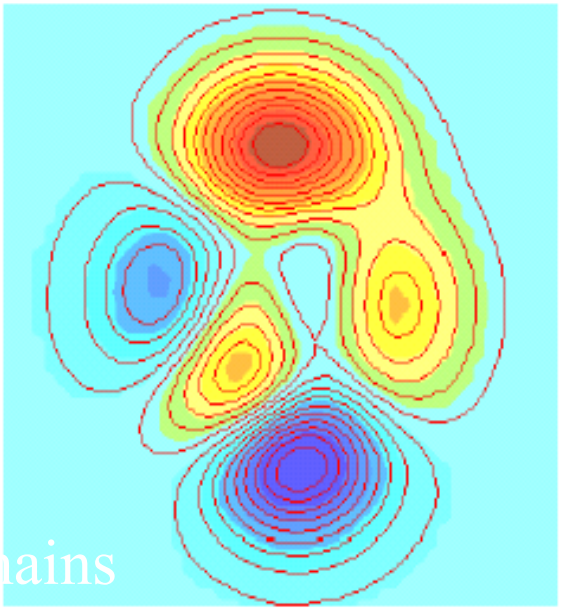




Example: Function optimisation (3)

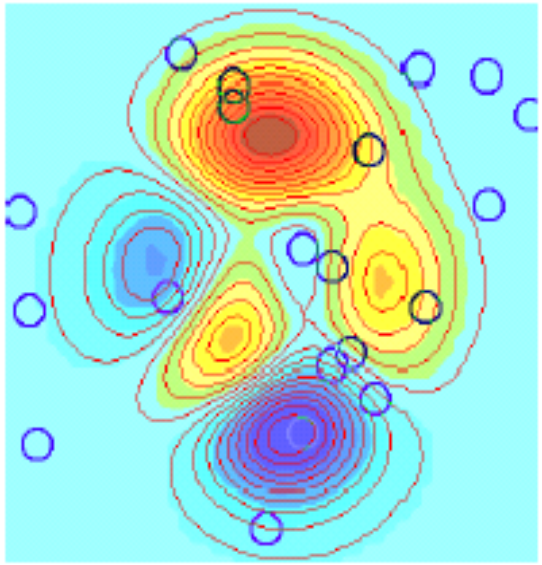
- $z = f(x, y) = 3*(1-x)^2*\exp(-(x^2) - (y+1)^2) - 10*(x/5 - x^3 - y^5)*\exp(-x^2-y^2) - 1/3*\exp(-(x+1)^2 - y^2)$.



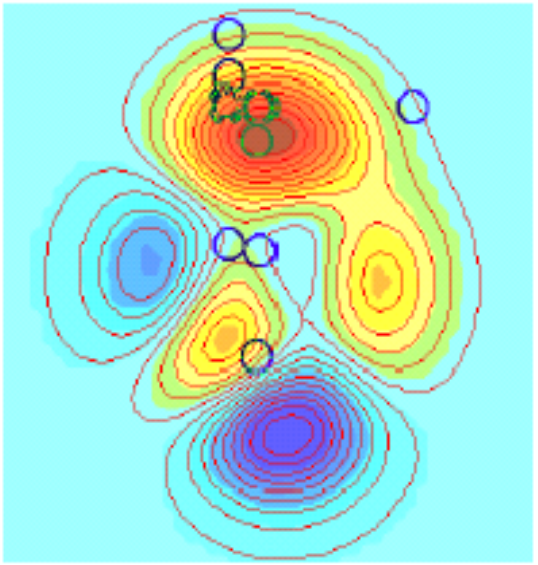


Example: Function optimisation (3)

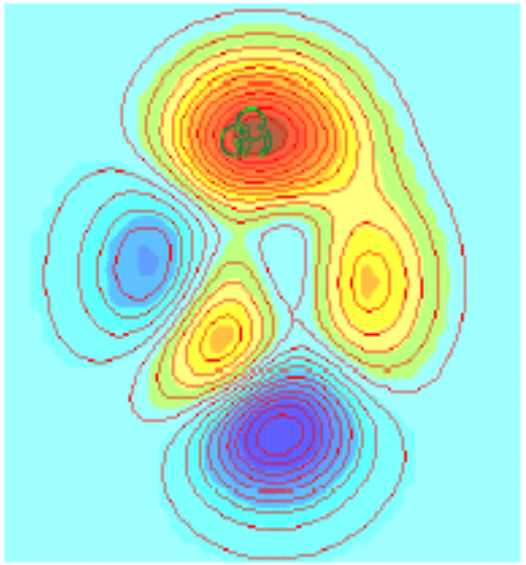
- GA process:



Initial population



5th generation



10th generation

Example: The (iterated) Prisoner's Dilemma game

- The 'Prisoner's dilemma game was invented by Merrill Flood & Melvin Dresher in the 50s
- Albert Tucker formalized the game with prison sentence rewards and gave it the name "prisoner's dilemma"
- Studied in game theory, economics, political science, cold war
- The story
  - Paula and Bruno arrested, no communication between them
  - They are offered a deal:
    - If **one** of them confesses & testifies against the other then gets suspended sentence while the other gets 5 years in prison
    - If **both** confess & testify against the other, they both get 4 years
    - If **none** of them confesses then they both get 2 years
  - What is the best strategy for maximising one's own payoff?

Example: The (iterated) Prisoner's Dilemma game

- Humans play against each other with the goal of maximixing one's payoff **in the long run**
- Winner strategy : TIT FOR TAT:
  - Cooperates as long as the other player does not defect
  - Defects on defection until the other player begins to cooperate again

  → Can a GA evolve a better strategy?

Encoding a strategy

Define a canonical order of cases

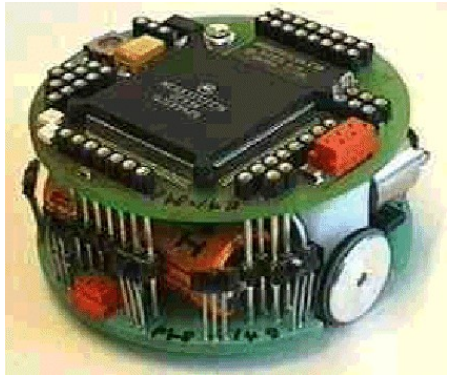
Case 1: CC	C	
Case 2: CD	D	encoding of TIT FOR TAT
Case 3: DC	C	
Case 4: DD	D	

- Results
  - For a environment formed by 8 human-designed strategies, a GA found a better strategy (i.e., highly adapted to *that* environment)

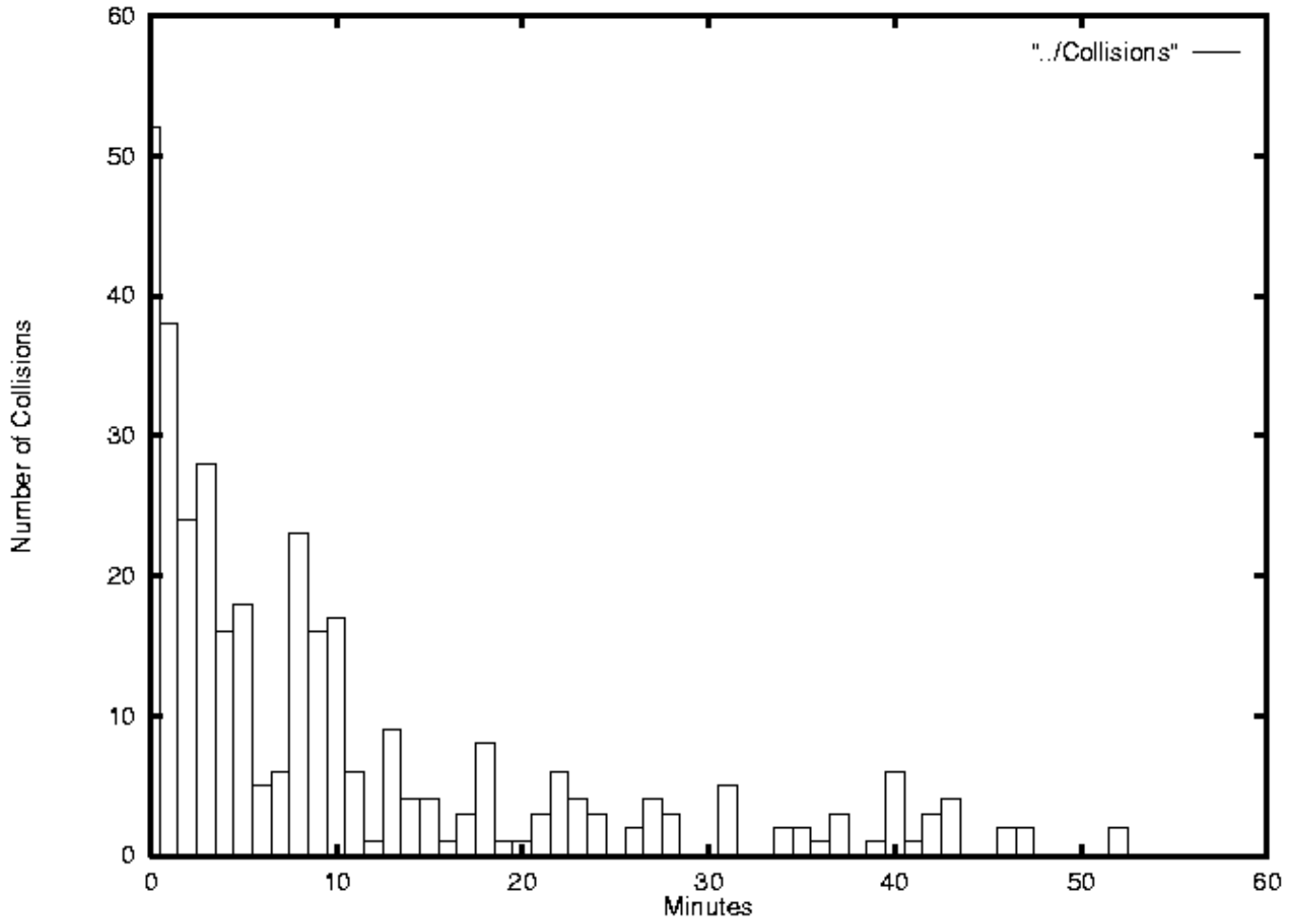
    [memory of 3 previous games, 40 runs, 50 generations each, population of 20, fitness: average score over all games played]
  - Experiments in changing environments: the evolving strategies played against each other: found strategies similar in essence with the winner human-designed strategy
  - Idealized model of evolution & co-evolution

Example: Evolving programs for a small mobile robot

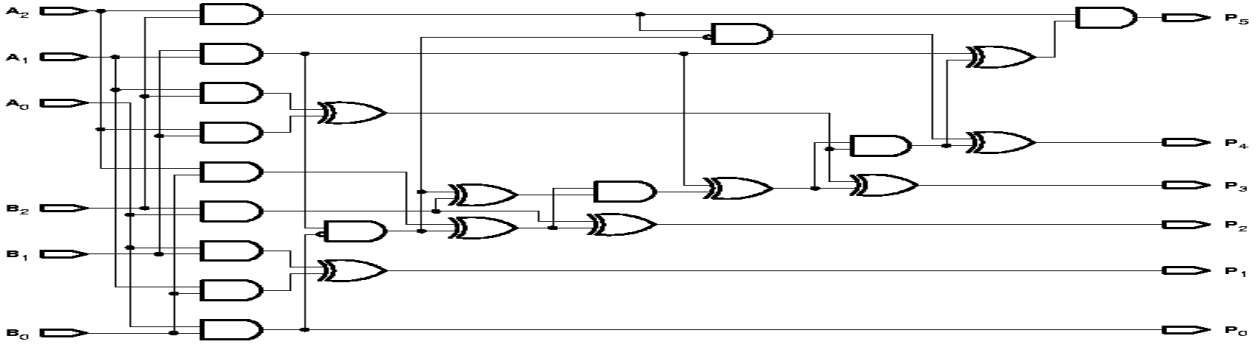
- Goal: obstacle avoidance
- Inputs from eight sensors on robot {s1-s8} with values in {0, ...,1023} (higher values mean closer obstacle)
- Output to two motors (speeds) with values in {0,15}.



Results

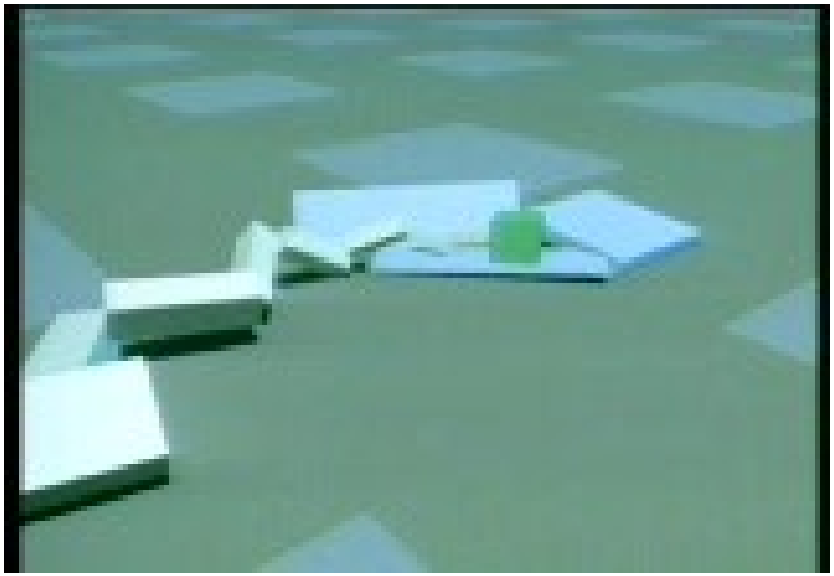


Example: Electronic Circuit Design



- Individuals are programs that transform an initial circuit to a final circuit by adding/subtracting components and connections
- Fitness: computed by simulating the circuit
- Population of 640,000 has been run on a parallel processor
- After 137 generations, the discovered circuits exhibit performance competitive with best human designs

Example: Karl Sims' virtual creatures



http://www.karlsims.com/evolved-virtual-creatures.html

Practical Issues I

- GAs: high flexibility and adaptability because of many options:
  - Problem representation
  - Genetic operators with parameters
  - Mechanism of selection
  - Size of the population
  - Fitness function
- Decisions are highly problem dependent
- Parameters not independent, you cannot optimize them one by one

Practical Issues (II)

Key ideas:

- The encoding should respect the schemata: it must allow discovery of small building blocks from which larger, more complete solutions can be formed
- The encoding should reflect functional interactions, as proximity on the genome (*linkage bias*)
- You should devise appropriate genetic operators:
  - all genotypes correspond to feasible solutions
  - genetic operators preserve feasibility

Practical Issues (III)

- Find **balance** between:
  - **Exploration** (new search regions)
  - **Exploitation** (exhaustive search in current region)
- Balance influenced by:
  - *Mutation, recombination*:
    - create individuals that are in new regions and/or fine tuning in current regions
  - *Selection*: focus on interesting regions
- Parameters can be adaptable, e.g. from high in the beginning (exploration) to low (exploitation), or even be subject to evolution themselves