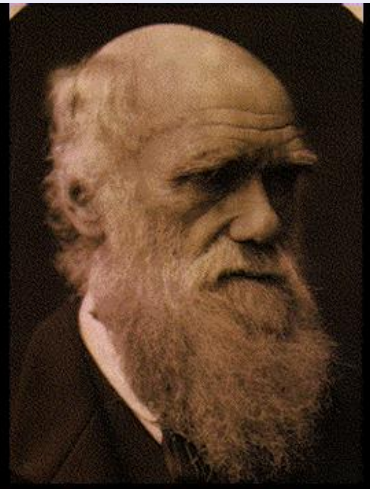
Fathers of evolutionary computing



Charles Darwin 1809 – 1882
"Father of the evolution theory"

Gregor Mendel 1822 – 1884
"Father of genetics"

John von Neumann 1903 – 1957
"Father of the computer"

Alan Mathison Turing 1912 – 1954
"Father of the computer"

Automated problem solving

Nature has always served as a source of inspiration for engineers and scientists.

Powerfull problem solvers known in nature is:

- The human brain that created "the wheel, New York, wars and so on" (after Douglas Adams Hitch-Hikers Guide)
- The evolution mechanism that created the human brain (after Darwins Origin of Species)

Automated problem solvers (algorithms):

- Neurocomputing
- Evolutionary computing (EC)

Alan Turing

Alan Turing in his 1948 paper "Intelligent Machinery":
"There is the genetical or evolutionary search by which a combination of genes is looked for, the criterion being the survival value".

In his 1950 paper "Computing Machinery and Intelligence":
We cannot expect to find a good child-machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications

- Structure of the child machine = Hereditary material
- Changes of the child machine = Mutations
- Natural selection = Judgment of the experimenter

History- founding the dialects of EC

- 1962, Bremermann: optimization through evolution and recombination
- 1964, Rechenberg introduces **evolution strategies** (ES)
- 1965, L. Fogel, Owens and Walsh introduce **evolutionary programming** (EP)
- 1975, Holland introduces **genetic algorithms** (GA)
- 1992, Koza introduces **genetic programming** (GP)

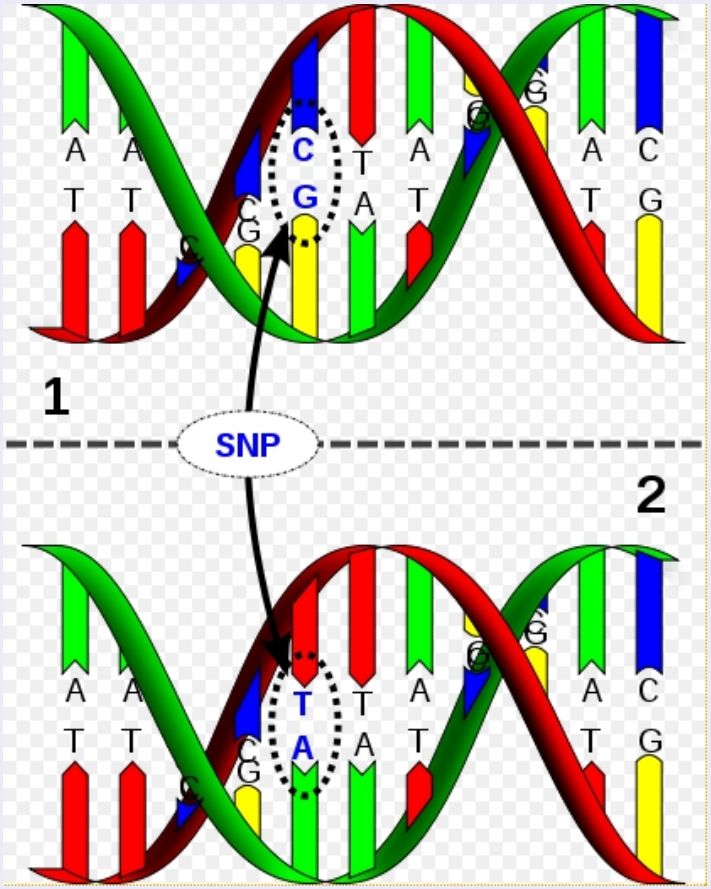
EC in the early 21st Century

- 3 major EC conferences, about 10 small related ones
- 3 scientific core EC journals
- 1000+ EC-related papers published in 2005 (estimate)
- EvoNet has over 150 member institutes
  - outdated by 2007
- uncountable (meaning: many) applications
- uncountable (meaning: ?) consultancy and R&D firms

Reference: A.E. Eiben and J.E. Smith (2007) *Introduction to Evolutionary Computing*, Springer, Natural Computing Series.

Genotype/phenotype

- The information required to build a living organism is coded in the DNA of that organism. The information may vary giving rise to different *genotypes*.
- Genotype determines *phenotype* (hair color, blood type, etc.)
- Genotype $\rightarrow$ phenotypic traits is a complex mapping
  - One genotype may affect many traits (pleiotropy)
  - Many genotypes may affect one trait (polygeny)
- Changes in the genotype may lead to changes in the organism (e.g., height, hair colour)

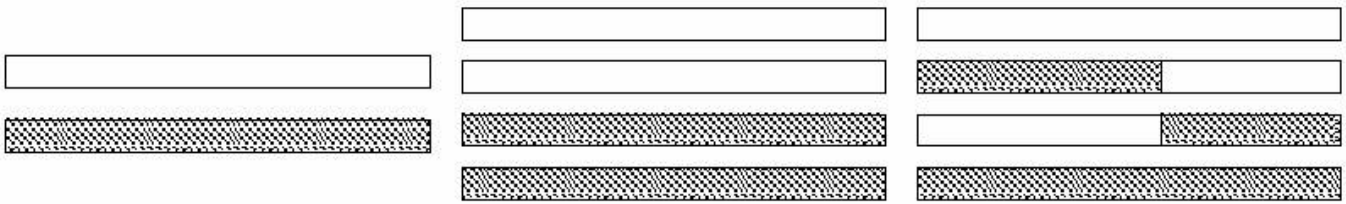


chromosomes

- Genes are encoded in strands of DNA called chromosomes
- In most cells, there are two copies of each chromosome (diploidy)
- Human DNA is organised into 23 pairs of chromosomes which together define the physical attributes of the individual
- *Gametes* (sperm and egg cells) contain 23 individual chromosomes rather than 23 pairs
- Gametes are formed by a special form of cell splitting called *meiosis*

Crossing over during meiosis

- Chromosome pairs align and duplicate
- Inner pairs link at a centromere and swap parts of themselves



- Outcome is one copy of maternal/paternal chromosome plus two entirely new combinations
- After crossing-over one of each pair goes into each gamete

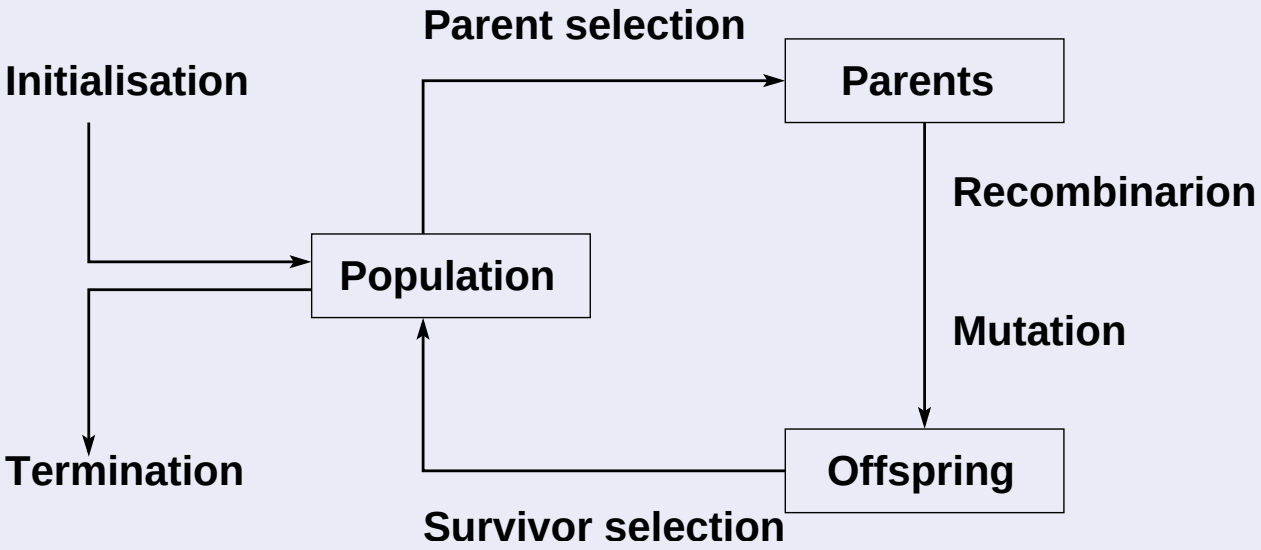
Mutation

- Occasionally some of the genetic material changes very slightly during meiosis(replication error)
- This means that the child might have genetic material information not inherited from either parent
  This can be
  - catastrophic: offspring in not viable (most likely)
  - neutral: new feature not influencing fitness
  - advantageous: strong new feature occurs
- Redundancy in the genetic code forms a good way of error checking

Pseudocode of an evolutionary algorithm(EA)

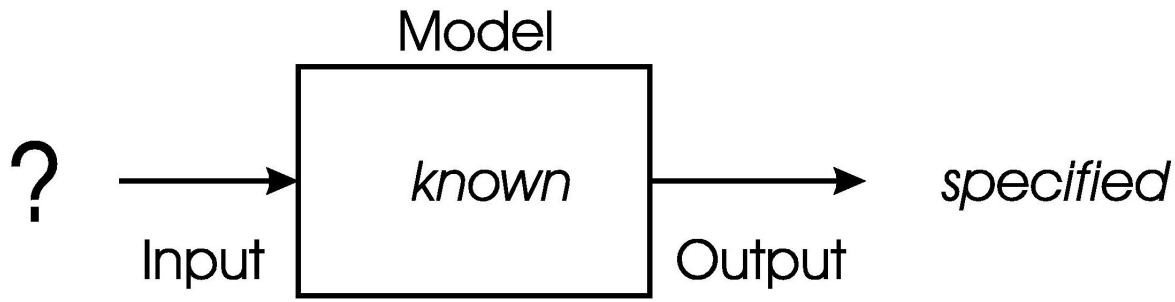
```
BEGIN
  INITIALIZE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT
    1. SELECT parents;
    2. RECOMBINE pairs of parents;
    3. MUTATE the resulting offspring;
    4. EVALUATE the new candidates;
    5. SELECT individuals for the next generation;
  UNTIL TERMINATION CONDITION is satisfied
END.
```

EA diagram


Parent selection
Initialisation
Parents
Recombinarion
Population
Mutation
Termination
Offspring
Survivor selection

Optimization

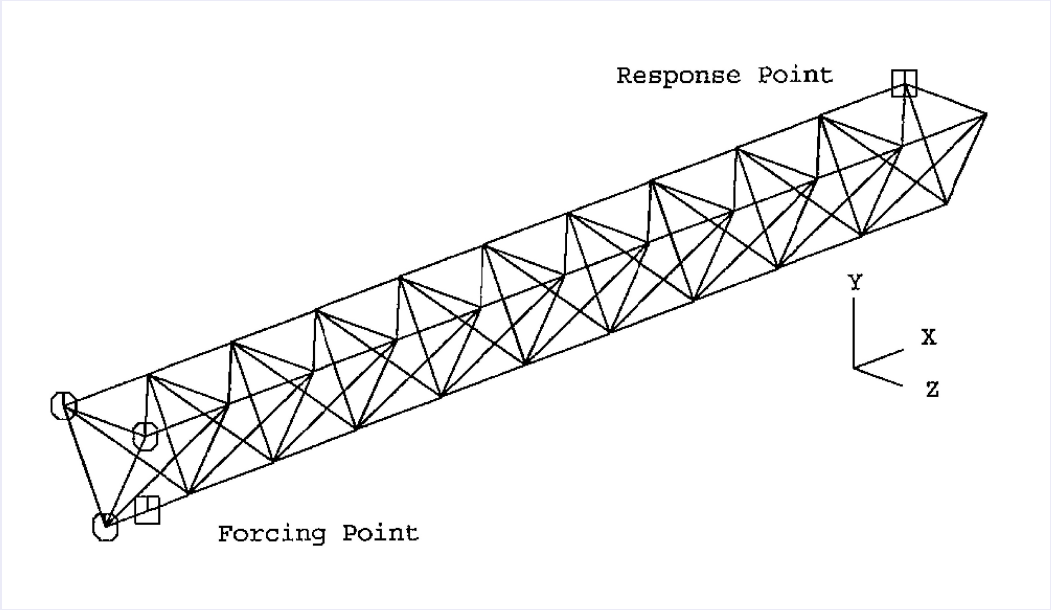
- We have a model of our system and seek inputs that give us a specified goal

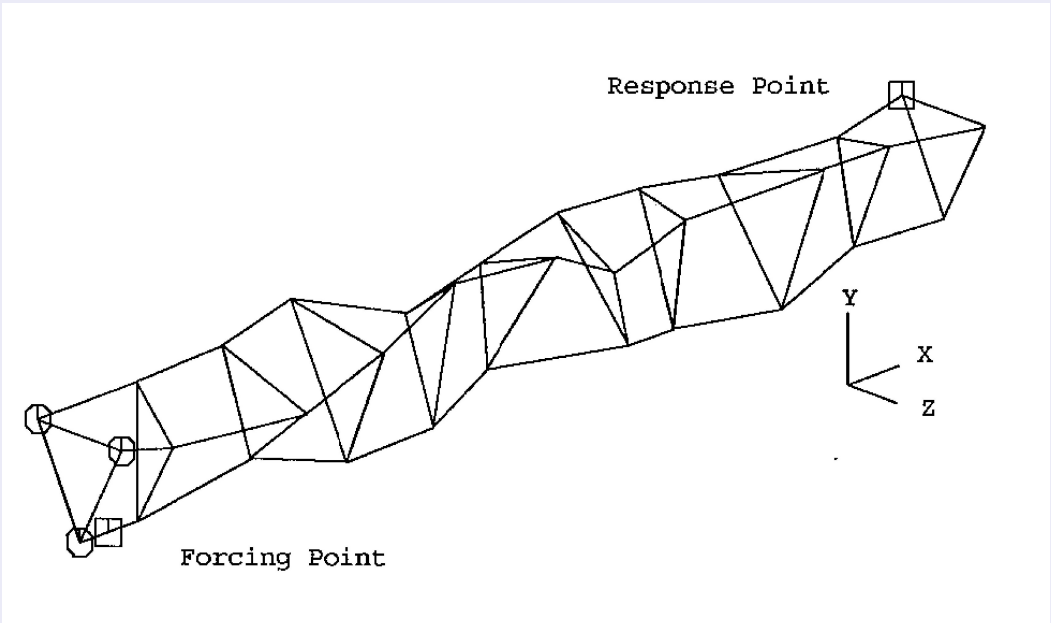


- e.g.
  - time tables for university, call center, or hospital
  - design specifications, etc etc

Example: Satellite structure

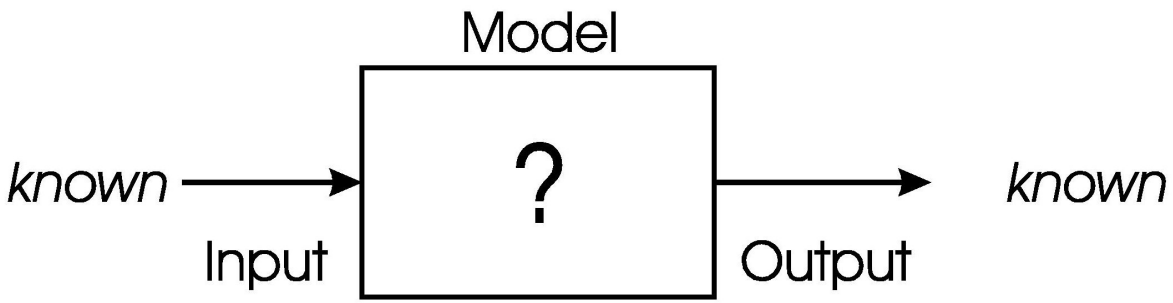
- Optimised satellite designs for NASA to maximize vibration isolation
- Evolving: design structures
- Fitness: vibration resistance





Modelling

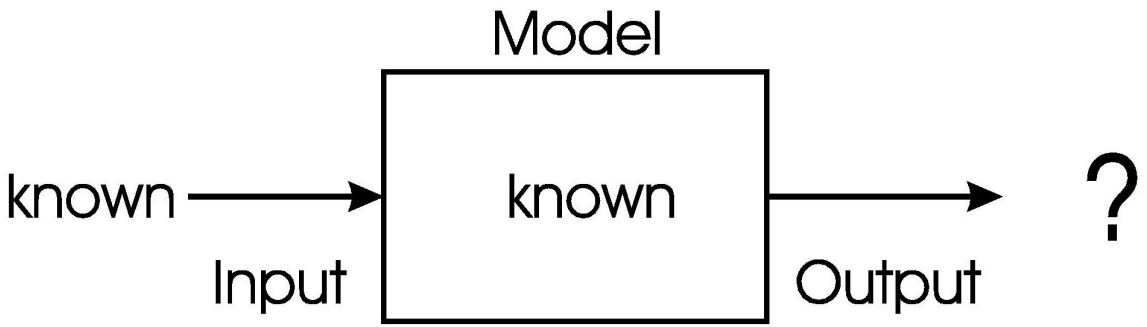
- We have corresponding sets of inputs & outputs and seek model that delivers correct output for every known input



- Evolutionary machine learning

Simulation

- We have a given model and wish to know the outputs that arise under different input conditions



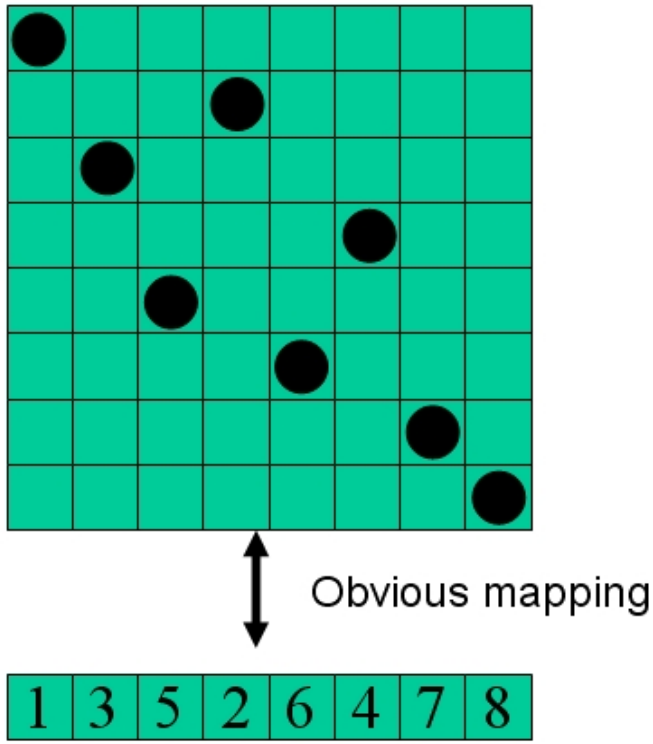
- Often used to answer "what-if" questions in evolving dynamic environments
- e.g. Evolutionary economics, Artificial Life

Example: the 8 queens problem

Place 8 queens on an 8x8 chessboard in such a way that they cannot check each other.

Representation

- Phenotype: A board configuration
- Genotype: A permutation



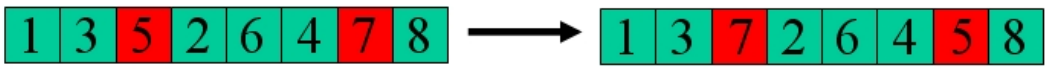
The 8 queens problem

Fitness evaluation

- Penalty of one queen: the number of queens she can check.
- Penalty of a configuration: the sum of the penalties of all queens.
- Note: penalty is to be minimized
- Fitness of a configuration: inverse penalty to be maximized

Mutation

- swapping values of two randomly chosen positions, or inverting a randomly chosen segment

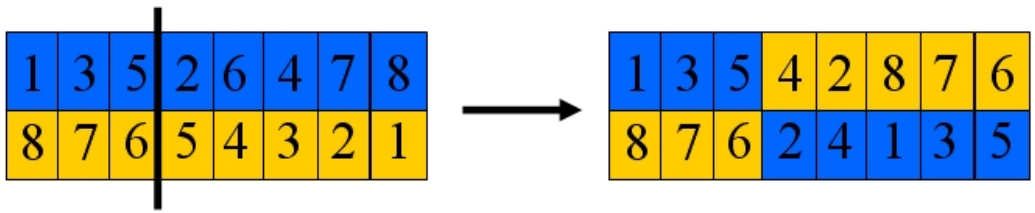


The 8 queens problem

Recombination

Combining two permutations into two new permutations:

- choose random crossover point
- copy first parts into children
- create second part by inserting values from other parent:
  - in the order they appear there
  - beginning after crossover point
  - skipping values already in child



The 8 queens problem

Selection

- Parent selection:
  Roulette wheel selection, for instance
- Survivor selection (replacement)
  When inserting a new child into the population, choose an existing member to replace by:
  - sorting the whole population by decreasing fitness
  - enumerating this list from high to low
  - replacing the first with a fitness lower than the given child

Magic squares

Given a 10x10 grid with a small 3x3 square in it.
Problem: arrange the numbers 1-100 on the grid such that

- all horizontal, vertical, diagonal sums are equal (505)
- a small 3x3 square forms a solution for 1-9

Evolutionary approach to solving this puzzle:

- Creating random begin arrangement
- Making N mutants of given arrangement
- Keeping the mutant (child) with the least error
- Stopping when error is zero

Software by M. Herdy, TU Berlin. Interesting parameters:

- Step1: small mutation, slow & hits the optimum
- Step10: large mutation, fast & misses ("jumps over" optimum)
- Mstep: mutation step size modified on-line, fast & hits optimum

Overview of dialects

Component or feature	EA Dialect			
	GA	ES	EP	GP
Typical problems	Combinatorial optimisation	Continuous optimisation	Optimisation	Modelling
Typical representation	Strings over a finite alphabet	Strings (vectors) of real numbers	Application specific often as in ES	Trees
Role of recombination	Primary variation operator	Important, but secondary	Never applied	Primary/only variation operator
Role of mutation	Secondary variation operator	Important, sometimes the only operator	The only variation operator	Secondary, sometimes not used at all
Parent selection	Random, biased by fitness	Random, uniform	Each individual creates one child	Random, biased by fitness
Survivor selection	Generational: n.a. all individuals replaced Steady-state: deterministic biased by fitness	Deterministic, biased by fitness	Random, biased by fitness	Random, biased by fitness

Table 15.1. Overview of the main EA dialects

GA quick overview

- Developed: USA in the 1970s
- Early names: J. Holland, K. DeJong, D. Goldberg
- Typically applied to:
  - discrete optimization
- Attributed features:
  - not too fast
  - good heuristic for combinatorial problems
- Special Features:
  - Traditionally emphasizes combining information from good parents (crossover)
  - many variants, e.g., reproduction models, operators

Overview

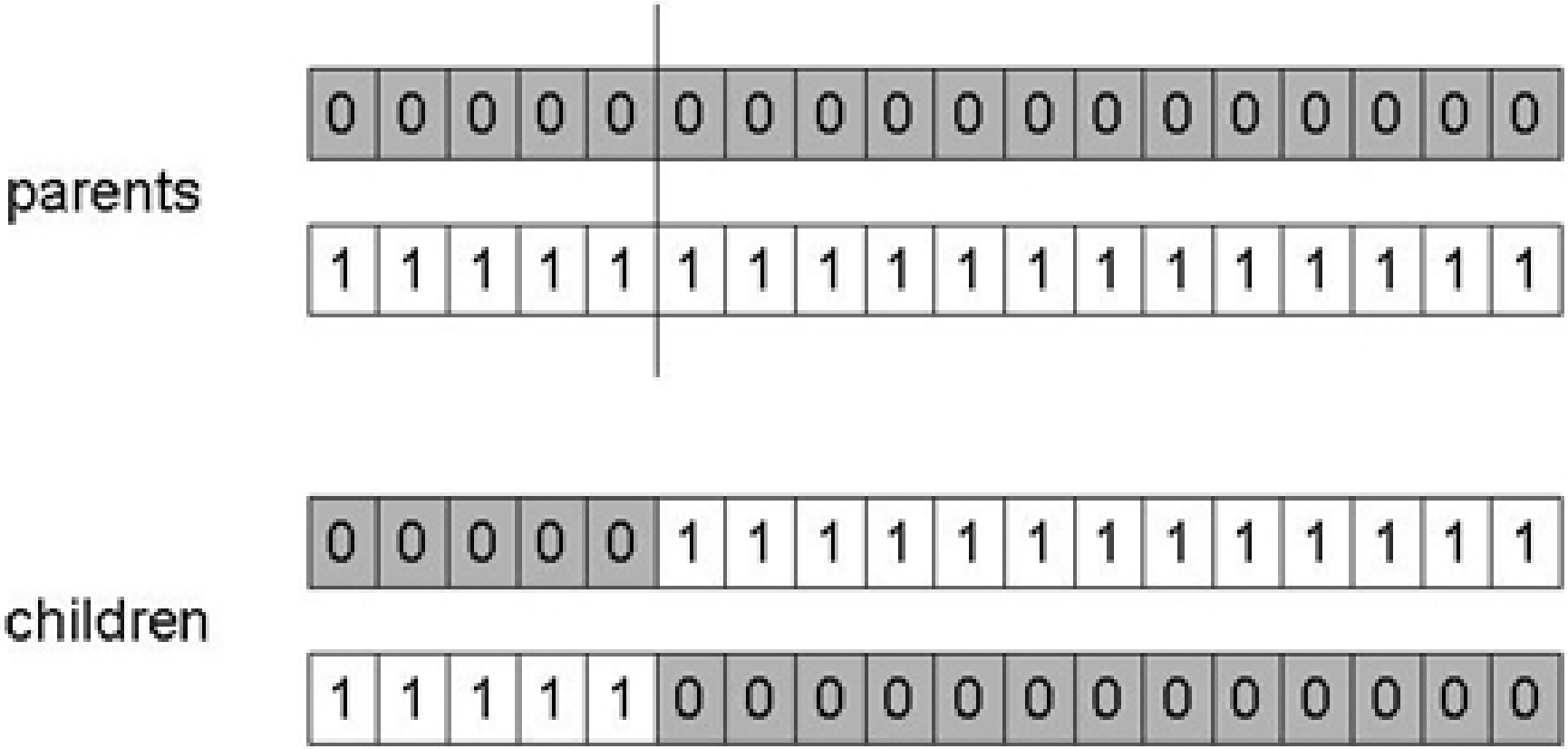
Representation	Binary strings
Recombination	N-point or uniform
Mutation	Bitwise bit-flipping with fixed probability
Parent selection	Fitness-Proportionate
Survivor selection	All children replace parents
Speciality	Emphasis on crossover

Reproduction cycle

1. Select with replacement parents for the mating pool (size of mating pool = population size)
2. Shuffle the mating pool. For each consecutive pair apply crossover with probability p_c , otherwise copy parents
3. For each offspring apply mutation (bit-flip with probability p_m independently for each bit)
4. Replace the whole population with the resulting offspring

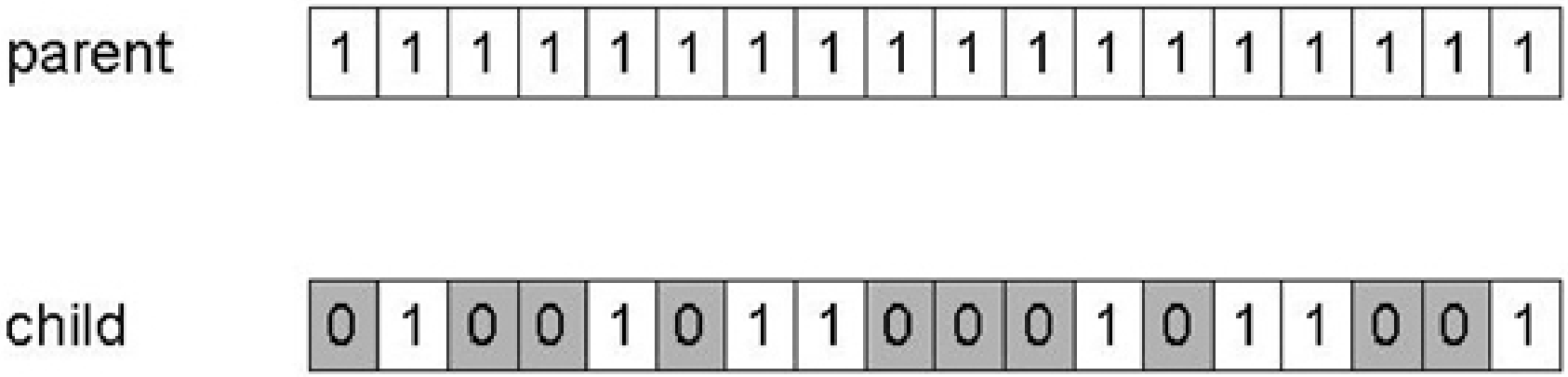
SGA operators: 1-point crossover

- Recombination with probability p_c typically in range (0.6, 0.9)
- Choose a random point on the two parents
- Split parents at this crossover point
- Create children by exchanging tails



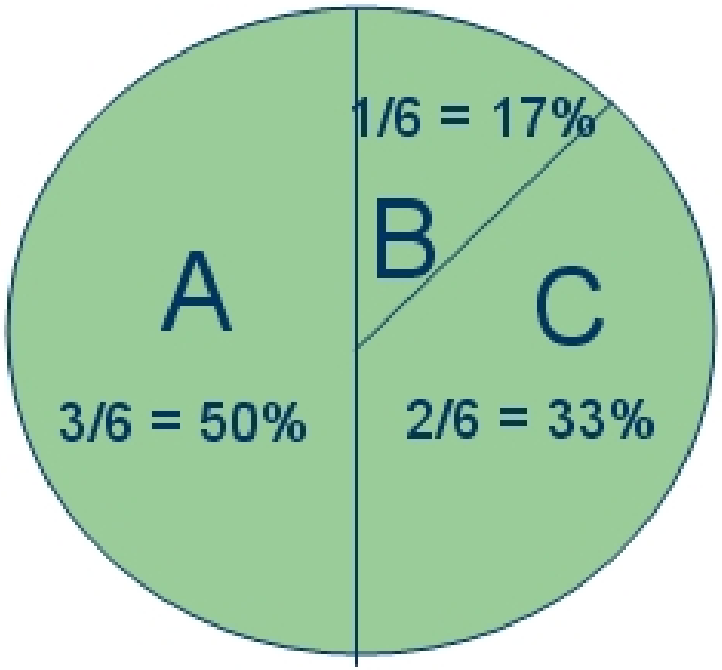
SGA operators: mutation

- Alter each gene independently with a probability p_m
- p_m is called the mutation rate - typically between 1/popSize and 1/chromosomeLength



SGA operators: Selection

- Main idea: better individuals get higher chance
- Chances proportional to fitness
- Implementation: roulette wheel technique
- Assign to each individual a part of the roulette wheel
- Spin the wheel n times to select n individuals



fitness(A) = 3

fitness(B) = 1

fitness(C) = 2

An example after Goldberg(1989)

- Simple problem: max x^2 over $\{0, \ldots, 31\}$
- GA approach:
  - Representation: binary code, e.g. $01101 \leftrightarrow 13$
  - Population size: 4
  - 1-point xover, bitwise mutation
  - Roulette wheel
  - Random initialisation
- We show one generational cycle done by hand

x^2 example: selection

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

x^2 example: crossover

String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

x^2 example: mutation

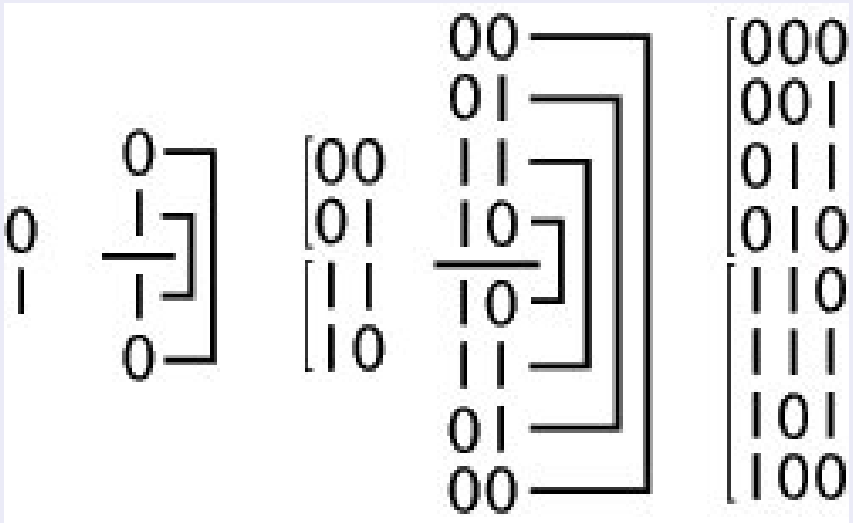
String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

Gray coding of integers

Gray coding is a mapping that means that small changes in the genotype cause small changes in the phenotype (unlike binary coding).

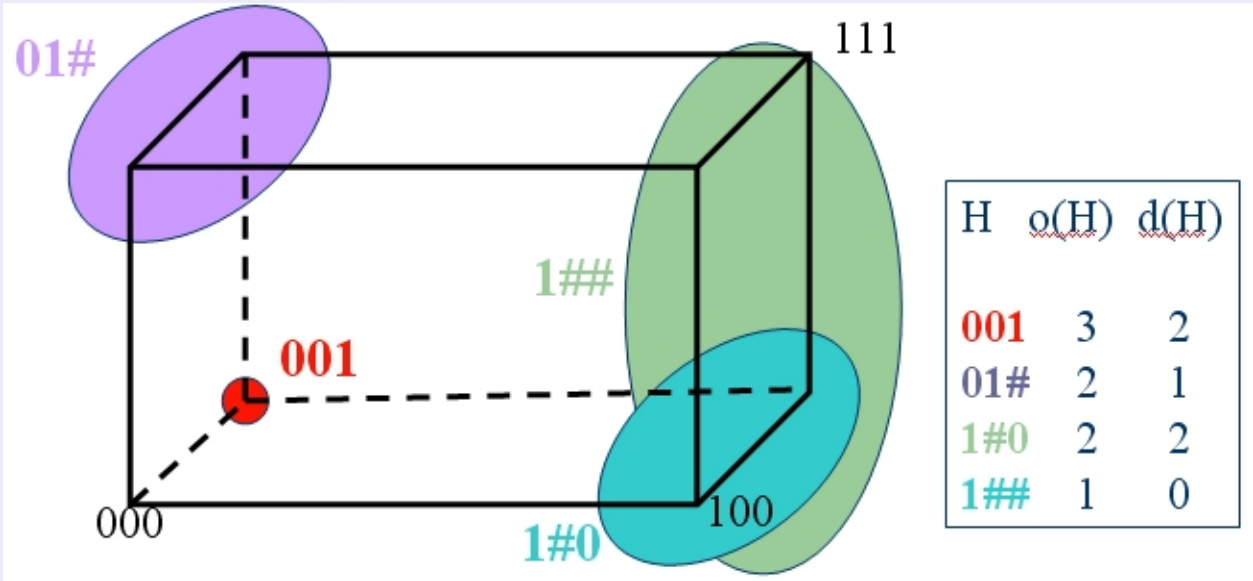
The binary-reflected Gray code for n bits can be generated recursively:

- list bits in reverse order and concatenate the reverse list onto the original list
- prefix the original bits with a binary 0 and prefix the reflected bits with a binary 1.



Example Schemata

- A schema (pl. schemata) is a string in a ternary alphabet (0,1,# = "dont care") representing a hyperplane within the solution space. E.g. $H_1 = 0001\#\#1\#\#0\#$ and $H_2 = \#\#1\#\#0\#\#$, etc.
- Two values can be used to describe schemata, the Order (number of defined positions), $o(H_1) = 6,\ o(H_2) = 2$.
  The Defining Length - length of sub-string between outmost defined positions, $d(H_1) = 9,\ d(H_2) = 3$.



H	o(H)	d(H)
001	3	2
01#	2	1
1#0	2	2
1##	1	0

Fitness proportionate selection

At generation t let

- $\langle f, t \rangle$ be the average fitness of the population
- $f(H, t)$ be the average fitness of the schema H
- $m(H, t)$ be the proportion of individuals in schema H

With Fitness Proportionate Selection, the next parent pool is expected to have a proportion of individuals in schema H given by

$$\tilde{m}(H, t+1) = \frac{m(H, t) f(H, t)}{\langle f, t \rangle}$$

Some of these may be disrupted by crossover/mutation and some new may be added. Hence, if $p_d(H)$ is the probability of disruption

$$m(H, t+1) \geq \frac{m(H, t) f(H, t)(1 - p_d(H))}{\langle f, t \rangle}$$

Schema theorem

Probability of disruption by mutation, when probability of allele mutation is p_m:

$$Pd(H, mutation) = 1 - (1 - p_m)^{o(H)} \approx o(H)p_m$$

Probability of disruption by one-point crossover(1X), when probability of crossover is p_c and length of gene is l:

$$Pd(H, 1X) \leq p_c \frac{d(H)}{l-1}$$

yielding **Holland's schema theorem**:

$$m(H, t+1) \geq m(H, t) \frac{f(H, t)}{\langle f, t \rangle} (1 - p_c \frac{d(H)}{l-1})(1 - o(H)p_m)$$

This means that short, low order, above average schemata receive exponentially increasing trials in subsequent generations of the classic genetic algorithm and below average schemata receive exponentially decreasing trials.

Markov chains

In general GA can be considered as simulating a Markov chain with state space being the genes of a population. HUGE dimension of state space. If X_t represents the generation at time t we clearly have

- X_t has a finite number of states
- The probability of being in any state at time $t+1$ depends only on the state at time t and is independent of t.

Has been used to provide convergence proofs, e.g. Eiben et al 1989 (almost sure convergence of GAs):

- IF the space is connected via variation operators AND
- IF selection is *elitist* AND
- 2 "trivial condition"

THEN for some n

$$P(\text{generation}(n) \text{ contains an optimum}) = 1$$

Elitist selection: The most fitted individual is ensured to survive.

GP overview

- Developed: USA in the 1990s
- Early names: J. Koza
- Typically applied to:
  - machine learning tasks (prediction, classification)
- Attributed features:
  - competes with neural nets and alike
  - needs huge populations (thousands)
  - slow
- Special:
  - non-linear chromosomes: trees, graphs
  - mutation possible but not necessary (disputed!)

Example: credit scoring

- Bank wants to distinguish good from bad loan applicants
- Model needed that matches historical data

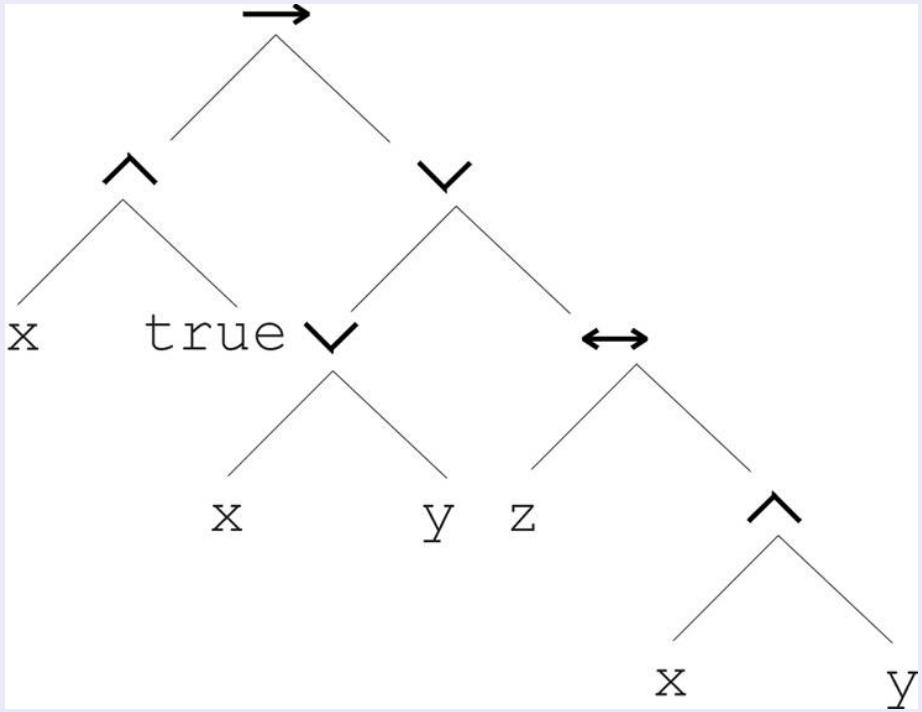
ID	No of children	Salary	Marital status	OK?
ID-1	2	90000	Married	0
ID-2	0	60000	Single	1
ID-3	1	80000	Divorced	1
...				

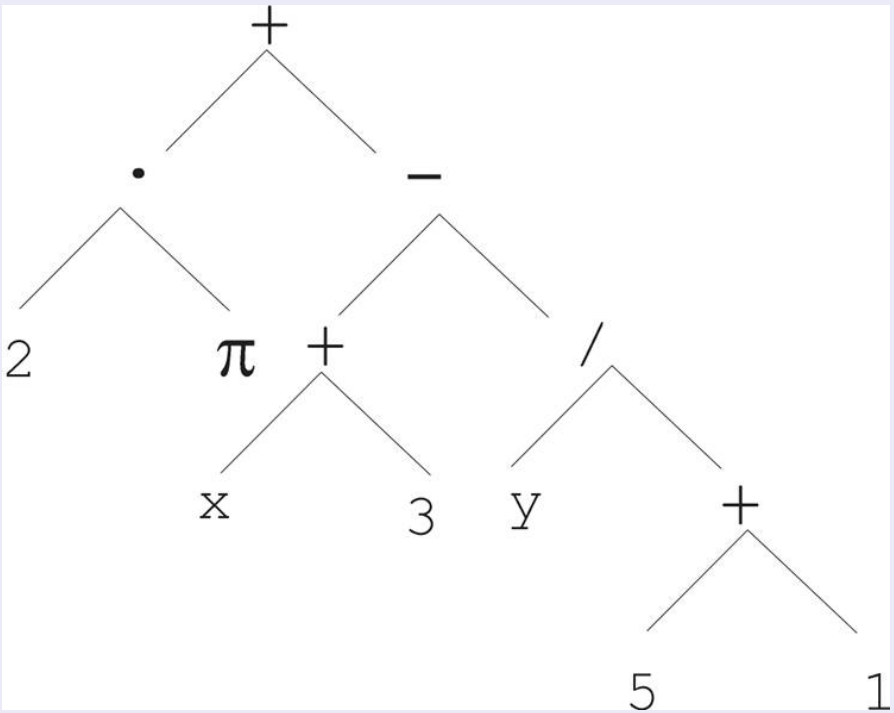
- A possible model:
  IF (NOC $= 2$) AND (S > 80000) THEN good ELSE bad
- Our search space (phenotypes) is the set of formulas
- Natural fitness of a formula: percentage of well classified cases of the model it stands for
- Natural representation of formulas (genotypes) is: parse trees

Tree based representation

Logical expression:

$$(x \wedge \text{true}) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$$





Arithmetic expression:

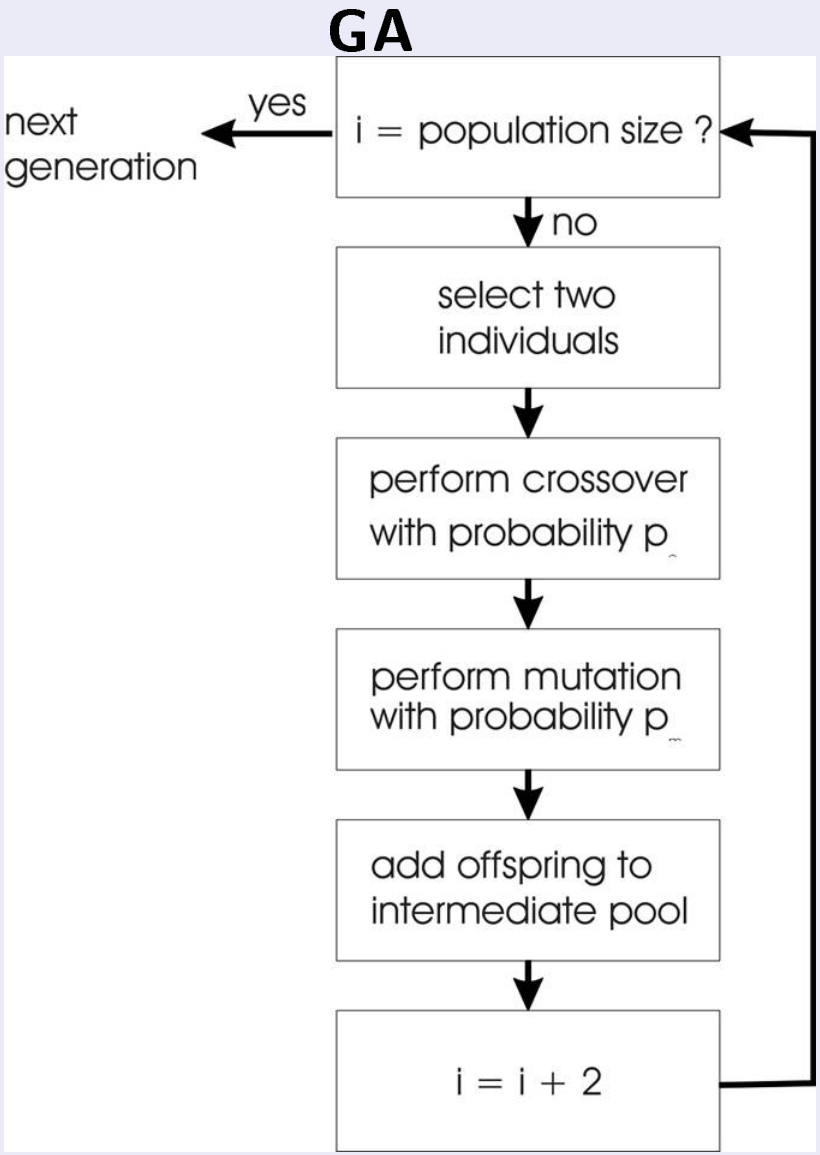
$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

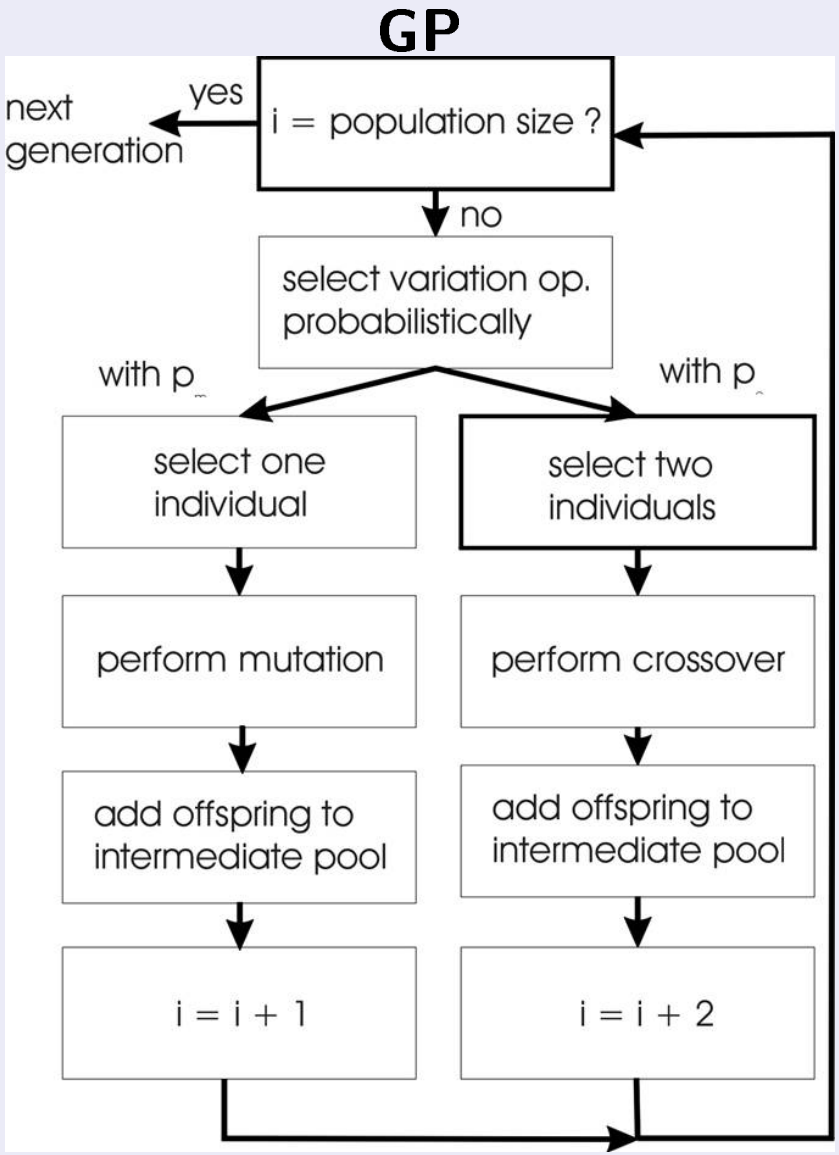
Symbolic regression

Given some points in $\mathbb{R}^2$: $(x_1, y_1), \ldots, (x_n, y_n)$.
Find function $f(x)$ s.t. for $i = 1, \ldots, n$ $f(x_i) = y_i$.
Possible GP solution:

- Representation by trees involving e.g
  - *Function set* $F = \{+, -, /, *, \sin, \cos\}$ for inner nodes
  - *Terminal set* $T = \mathbb{R} \cup x$ for terminal nodes
- Fitness is inverse to the squared error $\sum_1^n (y_i - f(x_i)^2$.

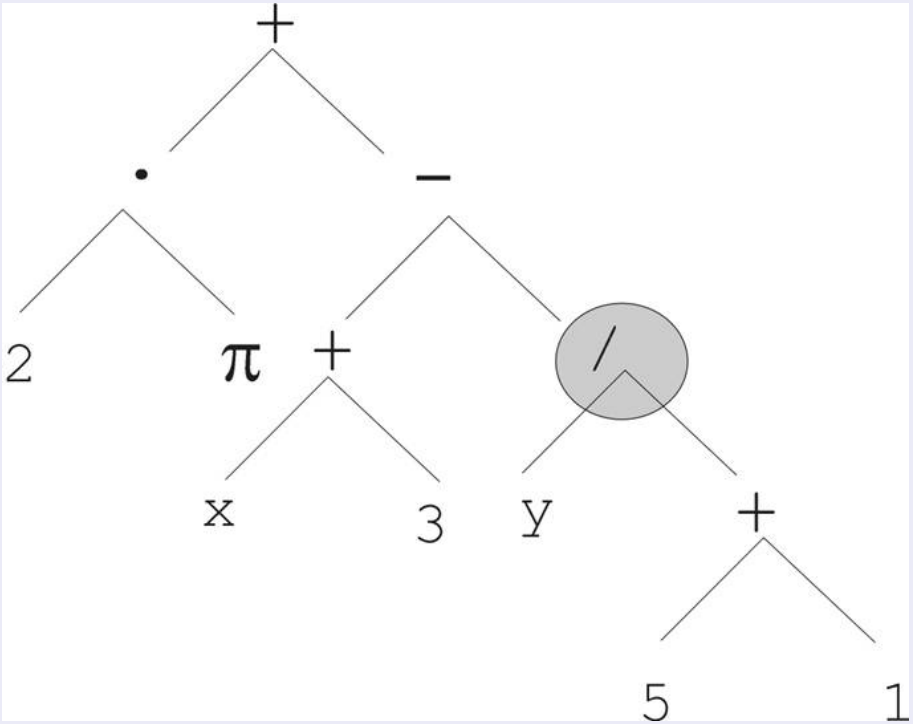
Offspring creation scheme

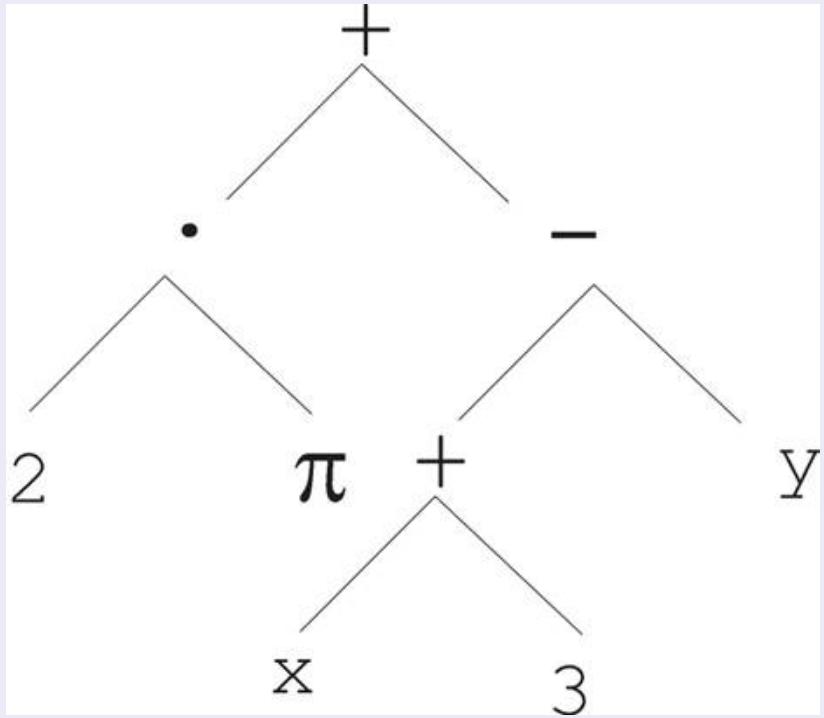




Mutation

Most common mutation: replace randomly chosen subtree by randomly generated tree.

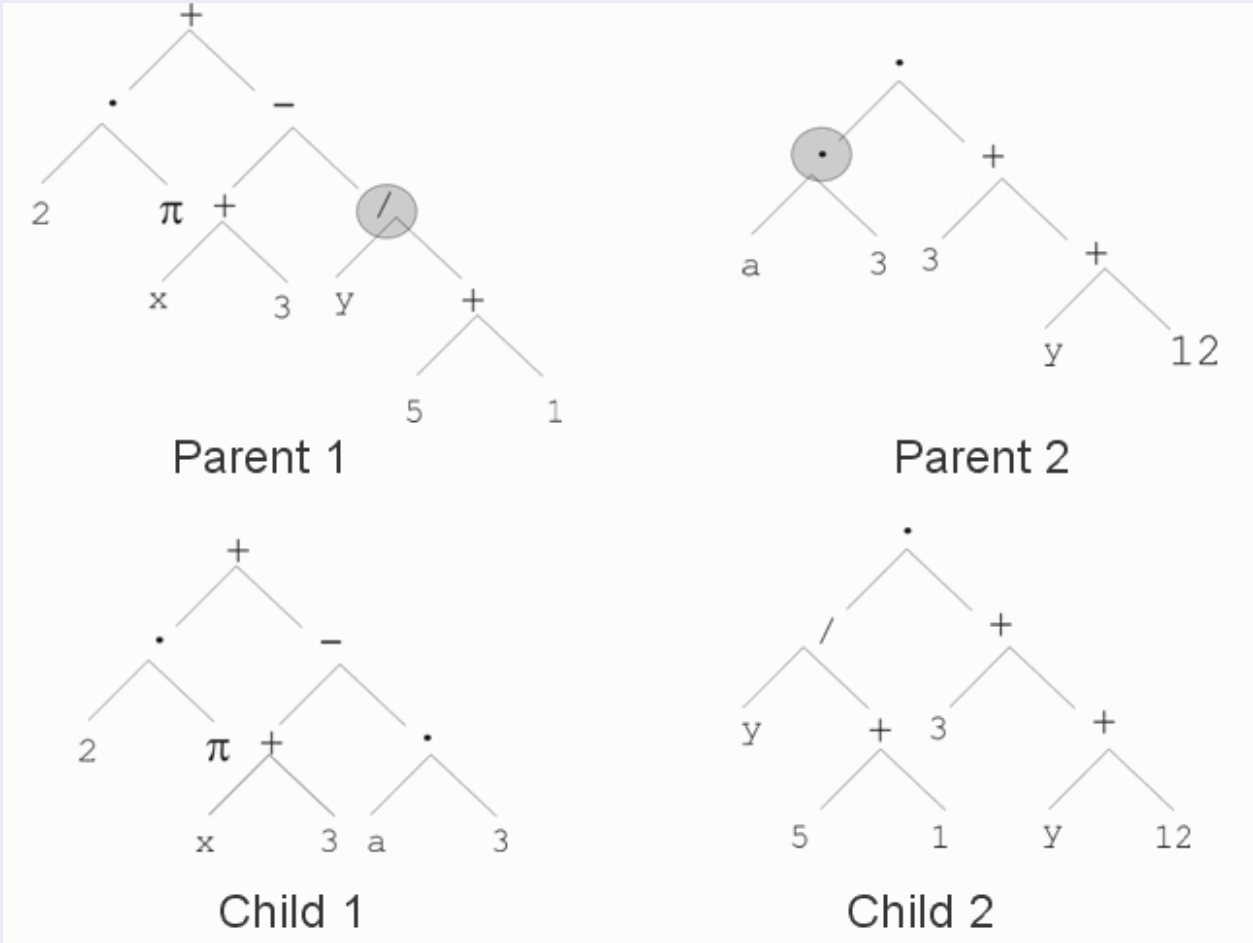




$$2\pi + ((x+3) - \frac{y}{5+1}) \rightarrow 2\pi + ((x+3) - y)$$

Crossover

Most common recombination: exchange two randomly chosen subtrees among the parents.



$2\pi + ((x + 3) - \frac{y}{5+1})$ & $3a(3 + (y + 12)) \rightarrow$
$2\pi + ((x + 3) - 3a)$ & $\frac{y}{5+1}(3 + (y + 12))$

Selection

- Parent selection typically fitness proportionate
- Over-selection in very large populations
  - rank population by fitness and divide it into two groups:
  - group 1: best x% of population, group 2 other (100-x)%
  - 80% of selection operations chooses from group 1, 20% from group 2
  - for pop. size = 1000, 2000, 4000, 8000 x = 32%, 16%, 8%, 4%
  - motivation: to increase efficiency, %'s come from rule of thumb

Bloat

- Bloat = "survival of the fattest", i.e., the tree sizes in the population are increasing over time
- Needs countermeasures, e.g.
  - Prohibiting variation operators that would deliver "too big" children
  - Parsimony pressure: penalty for being oversized

ES quick overview

- Developed: Germany in the 1970s
- Early names: I. Rechenberg, H.-P. Schwefel
- Typically applied to: numerical optimisation
- Attributed features:
  - fast
  - good optimizer for real-valued optimisation
  - relatively much theory

Representation	Real-valued vectors
Recombination	Discrete or intermediary
Mutation	Gaussian perturbation
Parent selection	Uniform random
Survivor selection	(μ, λ) or $(\mu + \lambda)$
Specialty	Self-adaptation of mutation step sizes

Example

- Task: minimimise $f : \mathbb{R}^n \to \mathbb{R}$
- Algorithm: "two-membered ES" using
  - Vectors from $\mathbb{R}^n$ directly as chromosomes
  - Population size 1
  - Only mutation creating one child
  - Greedy selection

Pseudocode

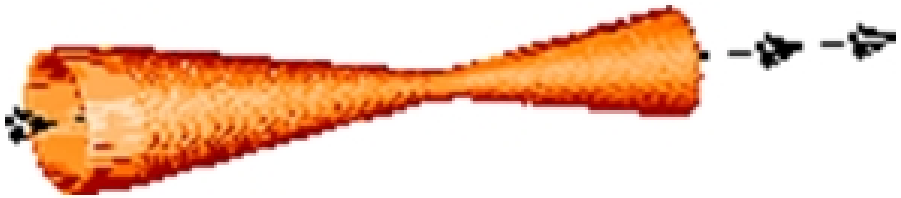
- Set $t = 0$
- Create initial point $x^t = (x_1^t, \ldots, x_n^t)$
- REPEAT UNTIL (TERMIN.COND satisfied)
  - Draw z_i from a normal distr. for all $i = 1, \ldots, n$
  - $y_i^t = x_i^t + z_i$
  - IF $f(x^t) < f(y^t)$ THEN $x^{t+1} = x^t$ ELSE $x^{t+1} = y^t$
  - Set $t = t + 1$

Example - mutation

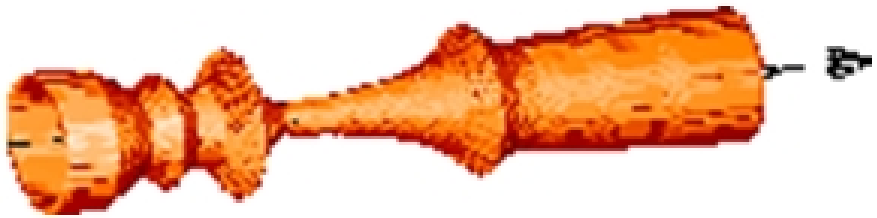
- z values drawn from normal distribution $N(0, \sigma)$
- variation σ is called mutation step size
- σ is varied on the fly by the "1/5 success rule".
- This rule resets σ after every k iterations by
  - $\sigma = \sigma/c$ if $p_s > 1/5$ $\sigma = \sigma c$ if $p_s < 1/5$ $\sigma = \sigma$ if $p_s = 1/5$
- where p_s is the % of successful mutations, $0.8 <\leq c \leq 1$.

A historical example: The jet nozzle experiment

Task: to optimize the shape of a jet nozzle
Approach: random mutations to shape + selection



Initial shape



Final shape

Lets see the movie

Correlated mutations

- Chromosomes: $\langle x_1, \ldots, x_n, \sigma_1, \ldots, \sigma_n, \alpha_1, \ldots, \alpha_k \rangle$ where $k = n(n-1)/2$
- The covariance matrix C is defined as:
  - $c_{ii} = \sigma_i^2$
  - $c_{ij} = 0$ if i and j are not correlated
  - $c_{ij} = \frac{1}{2}(\sigma_i^2 - \sigma_j^2) tan(2\alpha_{ij})$ if i and j are correlated
- Note the numbering/indices of the αs.

The mutation mechanism is then:

- $\sigma_i' = \sigma_i \exp(\tau' N(0,1) + \tau N_i(0,1))$
- $\alpha_j' = \alpha_j + \beta N(0,1)$
- $x' = x + N(0, C')$ where C' is the permuted covariance matrix
- $\tau' \sim 1/\sqrt{2n}$, $\tau \sim 1/\sqrt{2\sqrt{n}}$ and $\beta \sim 5^\circ$
- $\sigma_i' < \epsilon_0 \Rightarrow \sigma_i' = \epsilon_0$ and $|\alpha_j'| > \pi \Rightarrow \alpha_j' = \alpha_j' - 2\pi \mathtt{sign}(\alpha_j')$

Recombination

- Creates one child
- Acts per variable / position by either
  - Averaging parental values, or
  - Selecting one of the parental values
- From two or more parents by either:
  - Using two selected parents to make a child
  - Selecting two parents for each position anew

Parent selection

- Parents are selected by uniform random distribution whenever an operator needs one/some
- Thus: ES parent selection is unbiased - every individual has the same probability to be selected

Survivor selection

- Applied after creating λ children from the μ parents by mutation and recombination
- Deterministically chops off the "bad stuff"
- Selective pressure in ES is very high ($\lambda \approx 7\mu$ is the common setting)

Basis of selection is either:

- The set of parents and children: $(\mu + \lambda)$-selection, which is is an elitist strategy
- The set of children only: (μ, λ)-selection, which is often preferred for:
  - Better in leaving local optima
  - Better in following moving optima
  - Using the $+$ strategy bad σ values can survive too long if their host x is very fit

Convergence - runtime

- ES is a zero'th order method for optimization - also called derivative-free or direct search method
- Opposed to first order methods, which utilize the gradient of the objective function - e.g. quasi-Newton, steepest descent, and conjugate gradient methods
- Second order methods, which utilize the gradient and the Hessian - e.g. Newtons method.

Runtime result for the simple $ES(1+1)$:

- isotropically gaussian distributed mutation vectors
- *1/5-rule* used for their adaptation
- objective function is unimodal and monotone with respect to the euclidian distance from the minimum.

ensures asymptotically optimal runtime, $\Theta(n)$ steps/function evaluations are necessary and sufficient to halve the approximation error.

No Free Lunch Theorems

- No free lunch (NFL) Theorems apply to EC algorithms
  - Theorems imply there can be no universally efficient EC algorithm
  - Performance of one algorithm when averaged over all problems is identical to that of any other algorithm
- Suppose EC algorithm A applied to loss L
  - Let L^n denote lowest loss value from most recent N population elements after $n \geq N$ unique function evaluations
- Consider the probability that $L^n = \lambda$ after n unique evaluations of the loss:

$$P(L^n = \lambda | L, A)$$

NFL theorems state that the mean of above probabilities over all loss functions is independent of A

Wiki comment on NFL

- Intelligent design and the NFL theorem. See also: Neo-creationism
- The folkloric NFL theorem is often invoked by intelligent design proponents Robert J. Marks II and William Dembski as supporting intelligent design and Dembski's concept of specified complexity which he alleges is evidence of design.
- The scientific community has rejected both the notions of specified complexity and that the no free lunch theorem supports intelligent design.

Absolutely main reference

A.E. Eiben and J.E. Smith (2007) *Introduction to Evolutionary Computing*, Springer, Natural Computing Series.
With a lot of material "stolen" from
http://www.cs.vu.nl/ gusz/ecbook/ecbook-slides.html