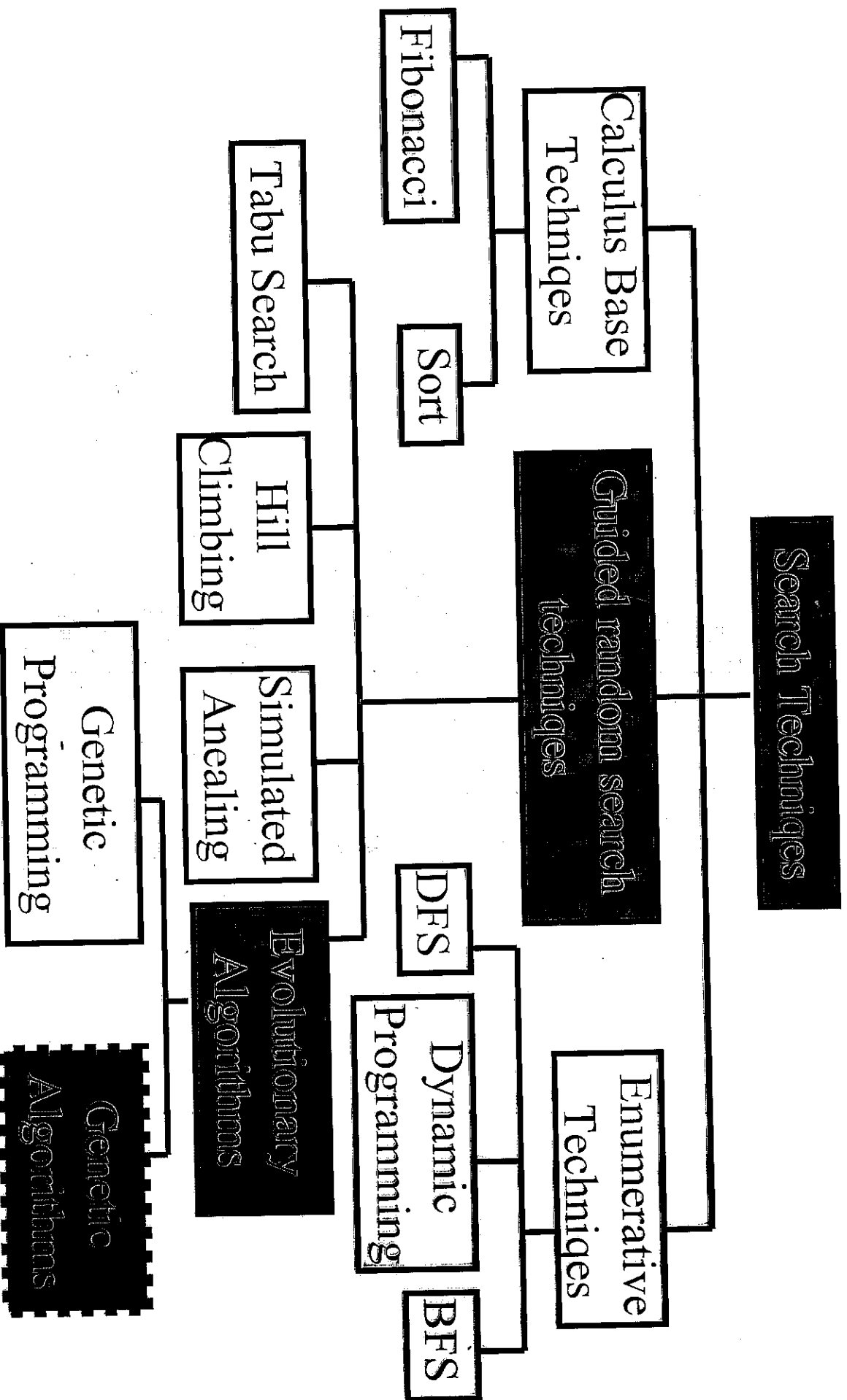


Classes of Search Techniques



The Metaphor

Genetic Algorithm	Nature
Optimization problem	Environment
Feasible solutions	Individuals living in that environment
Solutions quality (fitness function)	Individual's degree of adaptation to its surrounding environment

The Metaphor (cont)

Genetic Algorithm	Nature
A set of feasible solutions	A population of organisms (species)
Stochastic operators	Selection, recombination and mutation in nature's evolutionary process
Iteratively applying a set of stochastic operators on a set of feasible solutions	Evolution of populations to suit their environment

Simple Genetic Algorithm

produce an initial population of individuals

evaluate the fitness of all individuals

while termination condition not met **do**

 select fitter individuals for reproduction

 recombine between individuals

 mutate individuals

 evaluate the fitness of the modified individuals

 generate a new population

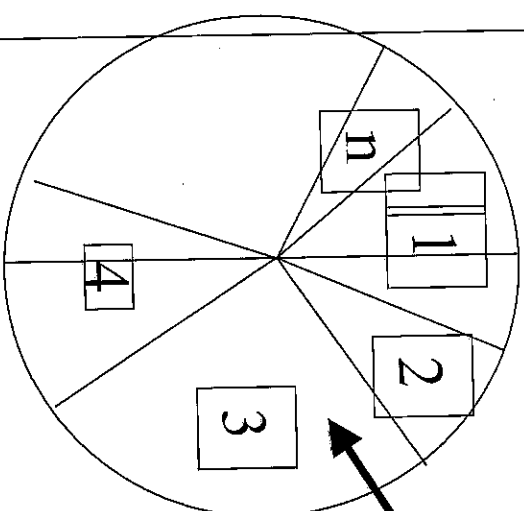
End while

Example (selection1)

Next we apply fitness proportionate selection with the roulette wheel method:

Individual i will have a $\frac{f(i)}{\sum_i f(i)}$ probability to be chosen

We repeat the extraction as many times as the number of individuals we need to have the same parent population size (6 in our case)



Area is Proportional to fitness value

Example (crossover²)

Before crossover:

$$s_1' = 11|11010101$$

$$s_2' = 11|10110101$$

$$s_5' = 01000|10011$$

$$s_6' = 11101|11101$$

After crossover:

$$s_1'' = 11|10110101$$

$$s_2'' = 11|11010101$$

$$s_5'' = 01000|11101$$

$$s_6'' = 11101|10011$$

Example (mutation1)

The final step is to apply random mutation: for each bit that we are to copy to the new population we allow a small probability of error (for instance 0.1)

Before applying mutation:

$$s_1'' = 11101 \ 0101$$

$$s_2'' = 1111 \ 1010$$

$$s_3'' = 11101 \ 11 \ 1$$

$$s_4'' = 0111000101$$

$$s_5'' = 0100011101$$

$$s_6'' = 11101100 \ 1$$

Components of a GA

A problem definition as input, and

- Encoding principles (gene, chromosome)
- Initialization procedure (creation)
- Selection of parents (reproduction)
- Genetic operators (mutation, recombination)
- Evaluation function (environment)
- Termination condition

Representation (encoding)

Possible individual's encoding

- Bit strings (0101 ... 1100)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E3 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)
- Program elements (genetic programming)
- ... any data structure ...

Representation (cont)

When choosing an encoding method rely on the following key ideas

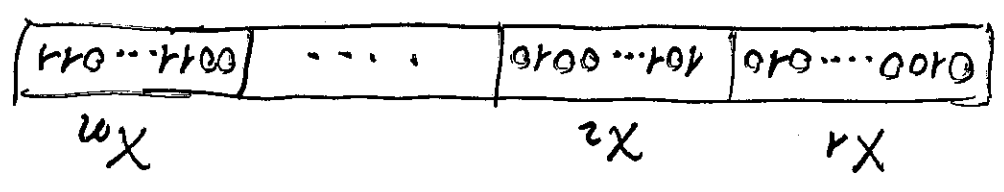
- Use a data structure as close as possible to the natural representation
- Write appropriate genetic operators as needed
- If possible, ensure that all genotypes correspond to feasible solutions
- If possible, ensure that genetic operators preserve feasibility

OPTIMISATION DE PARAMETRES CONTINUS: BINAIRE OU FLOATANT ?

inconvénient de la représentation binaire :
la précision est fonction du nombre de bits

Fonctions Multidimensionnelles : $f(x_1, x_2, \dots, x_n)$

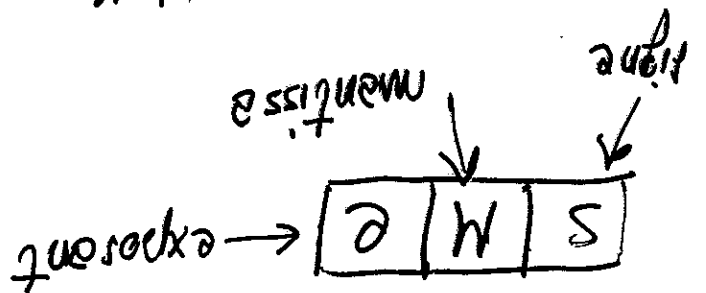
- chaînes très longues
 - contour
 - énormes espaces
- ↑ haute précision
← grands domaines



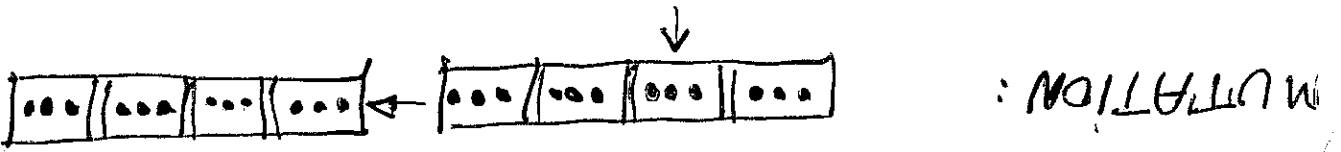
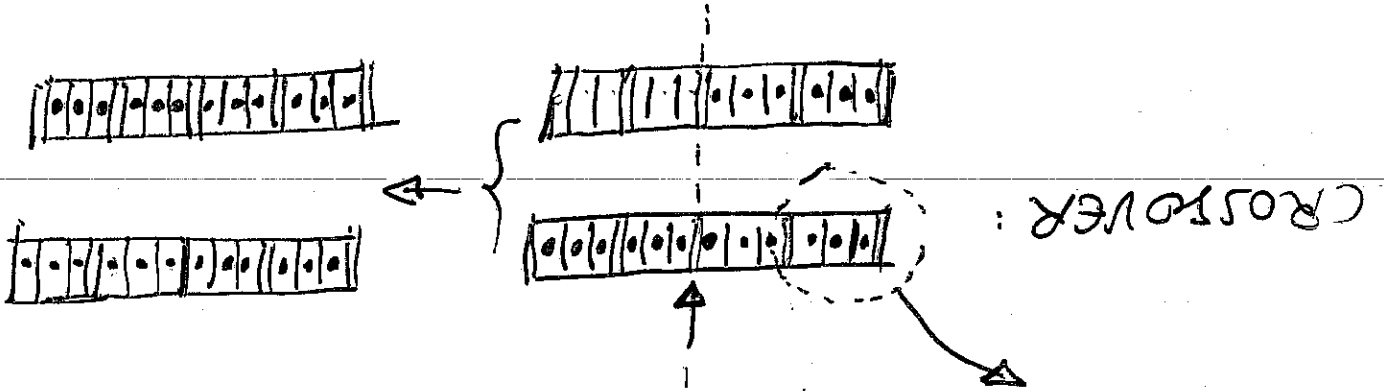
100 variables en $[-500, 500]$ avec une précision de 6 chiffres décimales : 300 bits

300
2
points

REPRÉSENTATION À VIRGULE FLOTTANTE



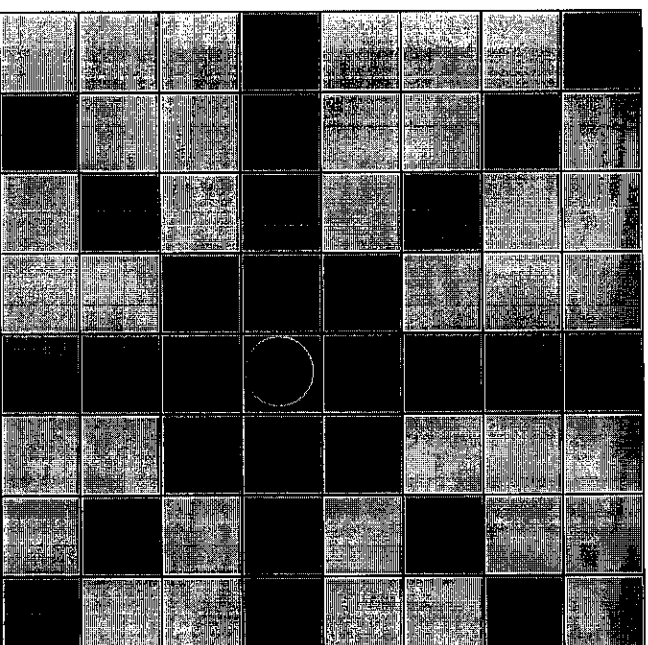
par ex, avec 16 bits on peut avoir une meilleure précision et l'on peut couvrir de plus amples domaines (160 bits pour 10 variables)



RESULTS: Représentation à virgule flottante plus rapide et plus précise

- nécessité d'opérateurs spécialisés
- support théorique quasi inexistant

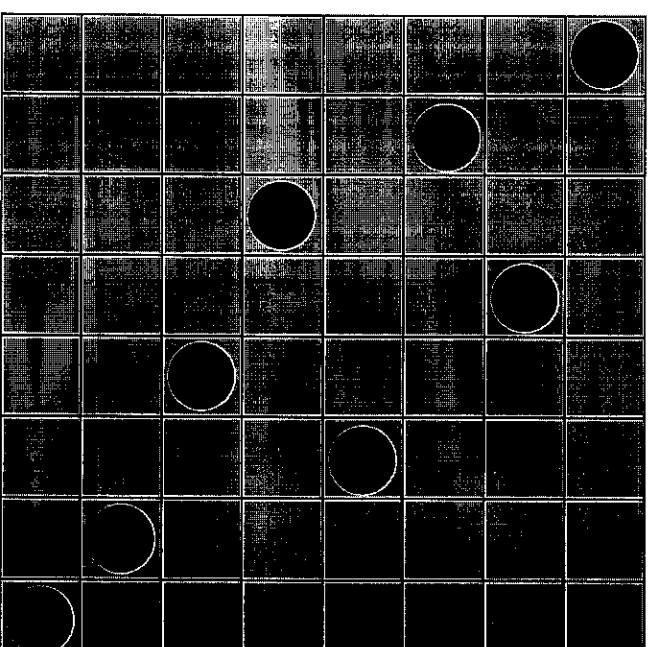
Example: the 8 queens problem



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

The 8 queens problem: representation

Phenotype:
a board configuration



Genotype:
a permutation of
the numbers 1 - 8



↕ Obvious mapping

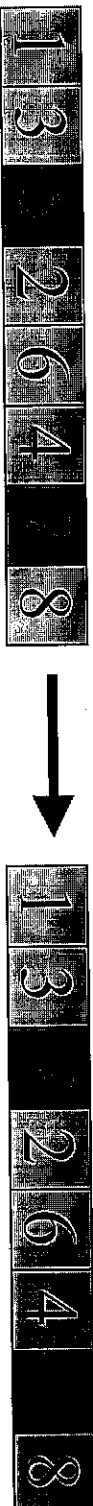
8 Queens Problem: Fitness evaluation

- 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

The 8 queens problem: Mutation

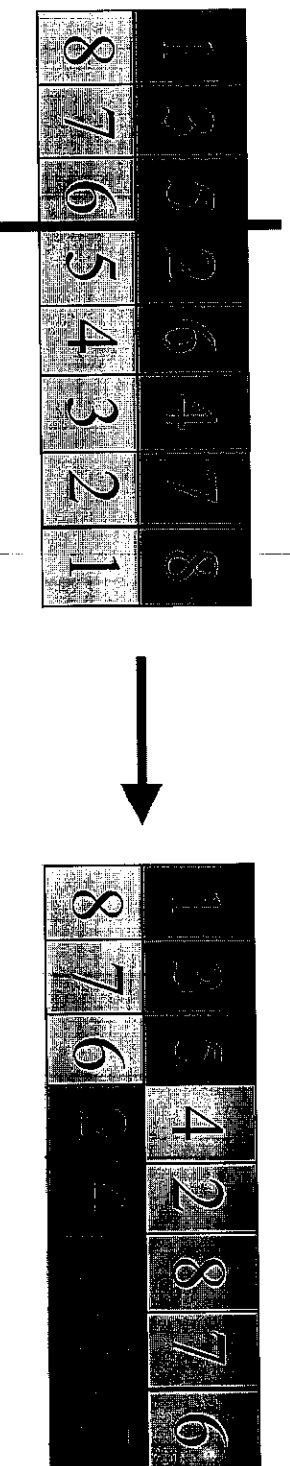
Small variation in one permutation, e.g.:

- swapping values of two randomly chosen positions,



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:**
 - Pick 5 parents and take best two to undergo crossover
- **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

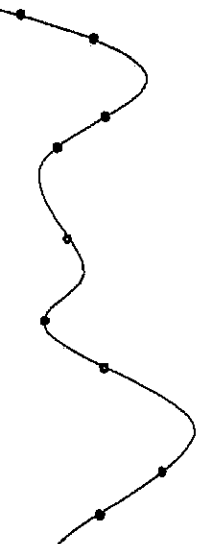
8 Queens Problem: summary

Representation	Permutations
Recombination	“Cut-and-crossfill” crossover
Recombination probability	100%
Mutation	Swap
Mutation probability	80%
Parent selection	Best 2 out of random 5
Survival selection	Replace worst
Population size	100
Number of Offspring	2
Initialisation	Random
Termination condition	Solution or 10,000 fitness evaluation

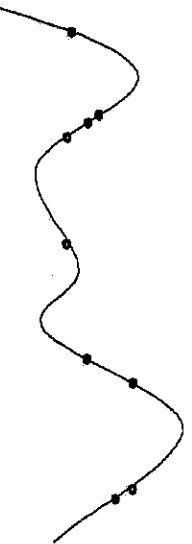
Note that is is ***only one possible***
set of choices of operators and parameters

Typical behaviour of an EA

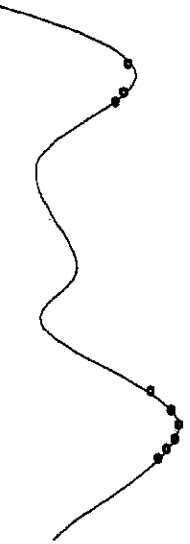
Phases in optimising on a 1-dimensional fitness landscape



Early phase:
quasi-random population distribution

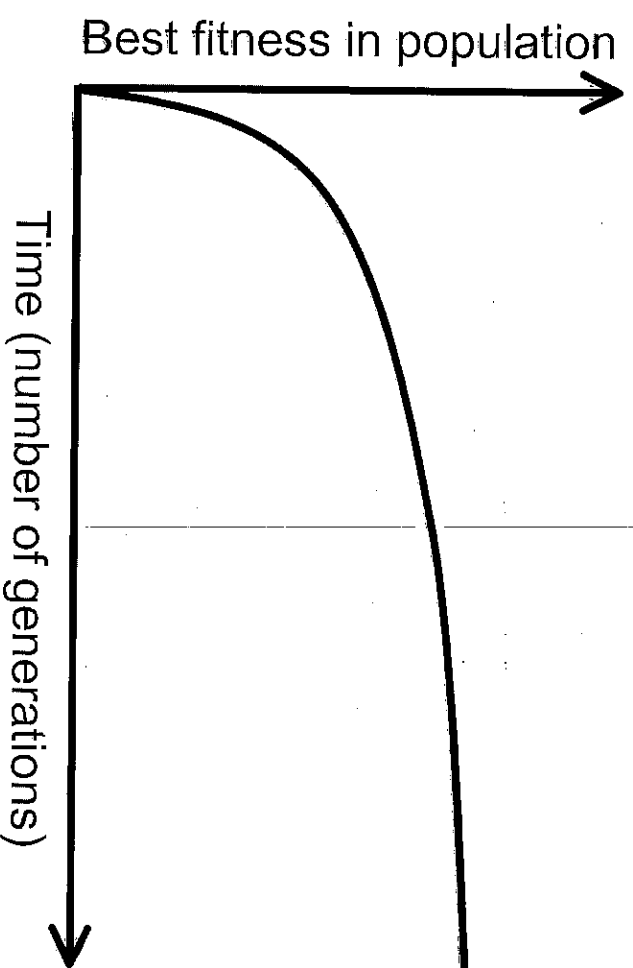


Mid-phase:
population arranged around/on hills



Late phase:
population concentrated on high hills

Typical run: progression of fitness



Typical run of an EA shows so-called “anytime behavior”

COMBINATORIAL OPTIMIZATION

A NUMBER PARTITIONING PROBLEM:

Given n numbers x_i ($i = 1, \dots, n$)
Partition them into K groups such that
differences between the group sums are
minimal.

K^n solutions, NP-hard problem

encoding:

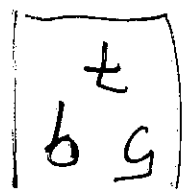
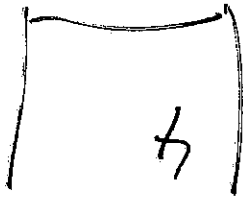
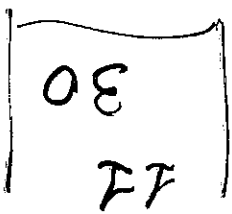
$$(p_1, p_2, \dots, p_n), \quad \forall p_i \in \{1, 2, \dots, K\}$$
$$i = 1, 2, \dots, n$$

group number encoding

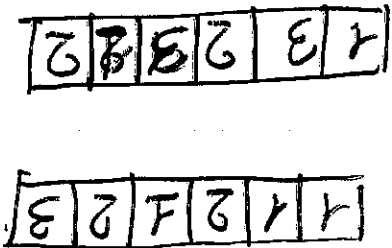
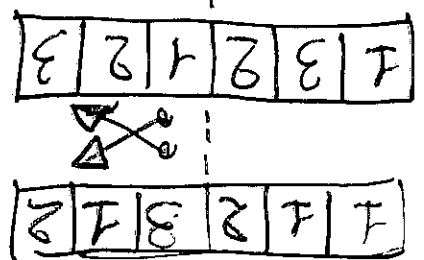
EXAMPLE:

$$X = \{5, 9, 4, 11, 7, 30\}, \quad k = 3$$

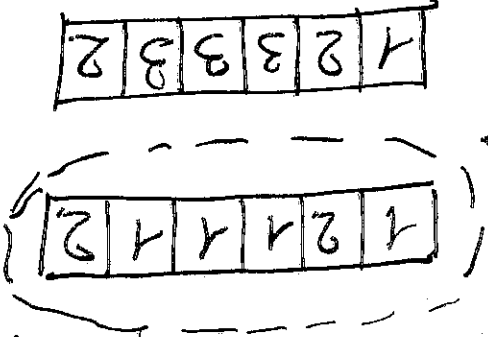
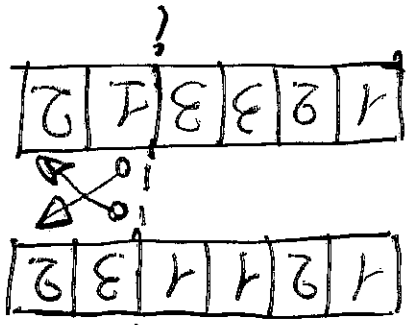
$$I = (1, 2, 3, 7) \quad \triangle$$



CROSSOVER:



HOWEVER:



BOX 3 NOT USED!

Selection

- Purpose: to focus the search in promising regions of the space
- Inspiration: Darwin's "survival of the fittest"
- Trade-off between *exploration* and *exploitation* of the search space

Next we shall discuss possible selection methods

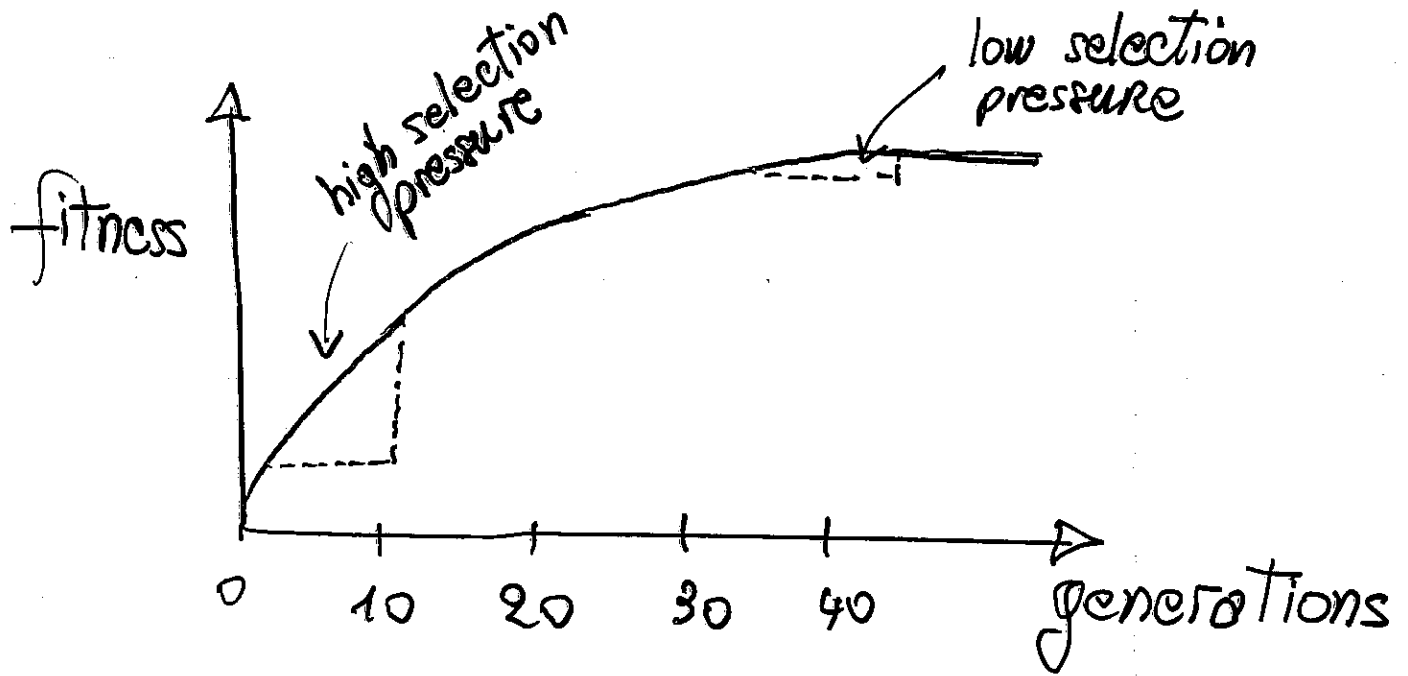
Fitness Proportionate Selection

- Derived by Holland as the optimal trade-off between exploration and exploitation

Drawbacks

- Different selection for $f_1(x)$ and $f_2(x) = f_1(x) + c$
- *Superindividuals* cause convergence (that may be premature)

SELECTION



- with fitness = evaluation selection pressure goes down with time (under fitness-proportionate selection)
- super individuals may take over the population

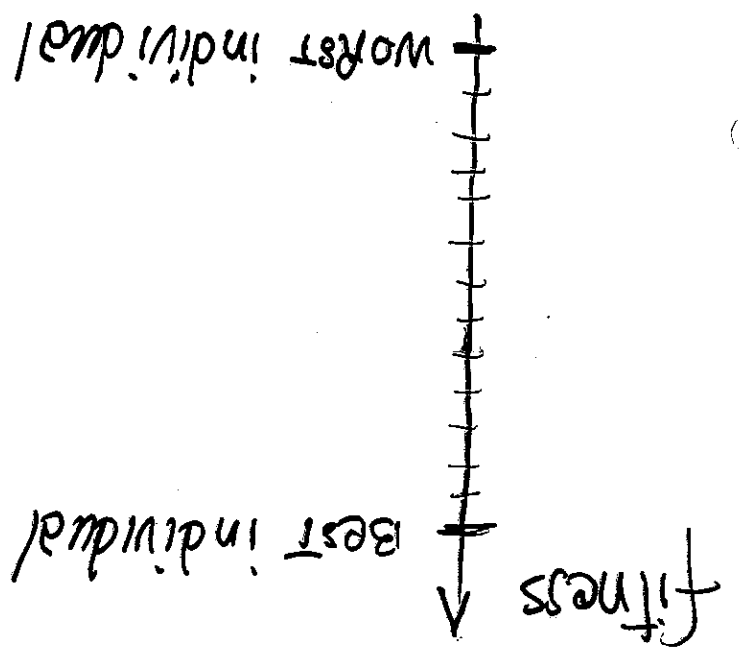
REMEDY: FITNESS SCALING

LINEAR: $f'_i = a f_i + b$

POWER LAW: $f'_i = f_i^k$

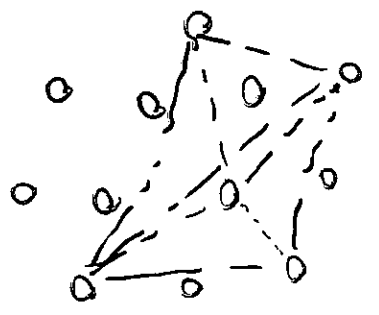
RANKING SELECTION

Prevents scaling
control selection Press.
violates schema Theor.

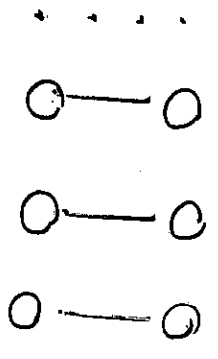


TOURNAMENT SELECTION

- select k individuals at a time
- The best one goes into the next generation
- Repeat Pop-size times



$k=2$ is often used:



Linear Ranking Selection

Based on sorting of individuals by decreasing fitness

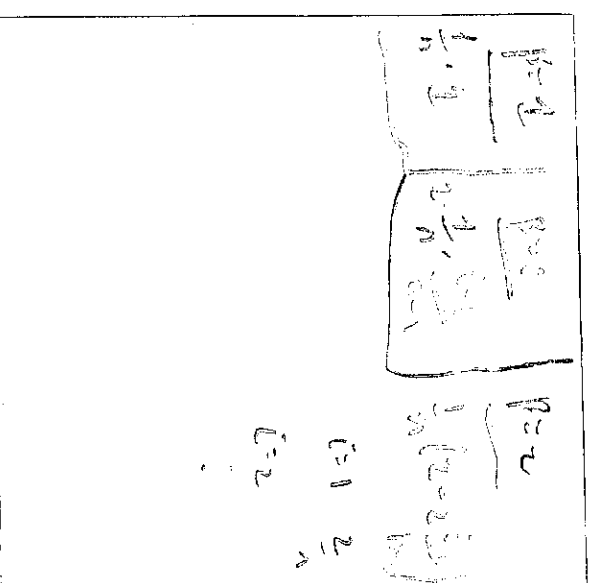
The probability to be extracted for the i th individual in the ranking is defined as

$$p(i) = \frac{1}{n} \left[\beta - 2(\beta - 1) \frac{i-1}{n-1} \right]$$

$$0 \leq \beta \leq 2$$

where β can be interpreted as the expected sampling rate of the best individual across n

independent & extractions with replacement



Local Tournament Selection

Extracts k individuals from the population with uniform probability (without re-insertion) and makes them play a “tournament”, where the probability for an individual to win is generally proportional to its fitness

Selection pressure is directly proportional to the number k of participants

Evolution strategies

Chapter 4

busz 2 Aihen
Tom Smith

ES quick overview

- Developed: Germany in the 1970's
- Early names: I. Rechenberg, H.-P. Schwefel
- Typically applied to:
 - numerical optimisation
- Attributed features:
 - fast
 - good optimizer for real-valued optimisation
 - relatively much theory
- Special:
 - self-adaptation of (mutation) parameters standard

ES technical summary tableau

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

Introductory example

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

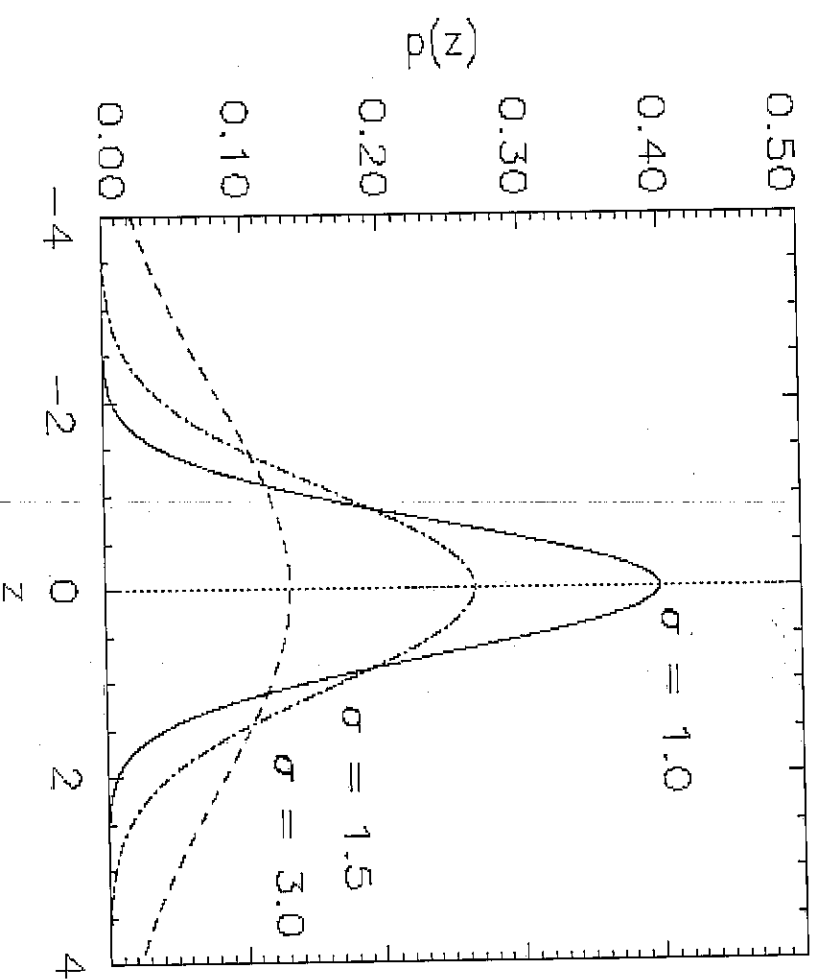
Introductory example: pseudocode

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

Introductory example: mutation mechanism

- z values drawn from normal distribution $N(\xi, \sigma)$
 - mean ξ is set to 0
 - 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 $p_s > 1/5$
 - $\sigma = \sigma \cdot c$ if $p_s < 1/5$
 - $\sigma = \sigma$ if $p_s = 1/5$
- where p_s is the % of successful mutations, $0.8 \leq c \leq 1$

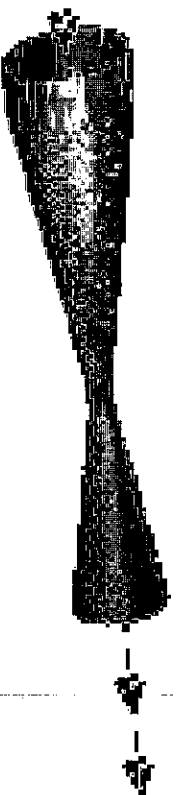
Illustration of normal distribution



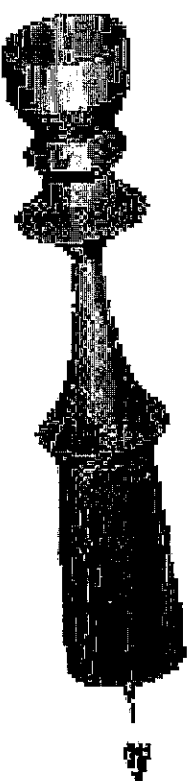
Another historical example: the jet nozzle experiment

Task: to optimize the shape of a jet nozzle

Approach: random mutations to shape + selection



Initial shape



Final shape

Representation

- Chromosomes consist of three parts:
 - Object variables: X_1, \dots, X_n
 - Strategy parameters:
 - Mutation step sizes: $\sigma_1, \dots, \sigma_{n_\sigma}$
 - Rotation angles: $\alpha_1, \dots, \alpha_{n_\alpha}$
- Not every component is always present
- Full size: $\langle X_1, \dots, X_n, \sigma_1, \dots, \sigma_n, \alpha_1, \dots, \alpha_k \rangle$
- where $k = n(n-1)/2$ (no. of i, j pairs)

Mutation

- Main mechanism: changing value by adding random noise drawn from normal distribution
- $x'_i = x_i + N(0, \sigma)$
- Key idea:
 - σ is part of the chromosome $\langle x_1, \dots, x_n, \sigma \rangle$
 - σ is also mutated into σ' (see later how)
- Thus: mutation step size σ is coevolving with the solution x

Mutate σ first

- Net mutation effect: $\langle x, \sigma \rangle \rightarrow \langle x', \sigma' \rangle$
- Order is important:
 - first $\sigma \rightarrow \sigma'$ (see later how)
 - then $x \rightarrow x' = x + N(0, \sigma')$
- Rationale: new $\langle x', \sigma' \rangle$ is evaluated twice
 - Primary: x' is good if $f(x')$ is good
 - Secondary: σ' is good if the x' it created is good
- Reversing mutation order this would not work

Mutation case 1: Uncorrelated mutation with one σ

- Chromosomes: $\langle X_1, \dots, X_n, \sigma \rangle$
- $\sigma' = \sigma \cdot \exp(\tau \cdot N(0, 1))$
- $X'_i = X_i + \sigma' \cdot N(0, 1)$
- Typically the “learning rate” $\tau \propto 1/n^{1/2}$
- And we have a boundary rule $\sigma' < \varepsilon_0 \Rightarrow \sigma' = \varepsilon_0$

Mutation case 2: Uncorrelated mutation with n σ 's

- Chromosomes: $\langle X_1, \dots, X_n, \sigma_1, \dots, \sigma_n \rangle$
- $\sigma'_i = \sigma_i \cdot \exp(\tau' \cdot N(0, 1) + \tau \cdot N_i(0, 1))$
- $X'_i = X_i + \sigma'_i \cdot N_i(0, 1)$
- Two learning rate parameters:
 - τ' overall learning rate
 - τ coordinate wise learning rate
- $\tau \propto 1/(2n)^{1/2}$ and $\tau \propto 1/(2n^{1/2})^{1/2}$
- And $\sigma'_i < \epsilon_0 \Rightarrow \sigma'_i = \epsilon_0$

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

Names of recombinations

	Two fixed parents	Two parents selected for each i
$z_i = (x_i + y_i)/2$	Local intermediary	Global intermediary
z_i is x_i or y_i chosen randomly	Local discrete	Global discrete

Survivor selection cont'd

- $(\mu+\lambda)$ -selection is an elitist strategy
- (μ,λ) -selection can “forget”
- Often (μ,λ) -selection is preferred for:
 - Better in leaving local optima
 - Better in following moving optima
 - Using the + strategy bad σ values can survive in $\langle x,\sigma \rangle$ too long if their host x is very fit
- Selective pressure in ES is very high ($\lambda \approx 7 \cdot \mu$ is the common setting)