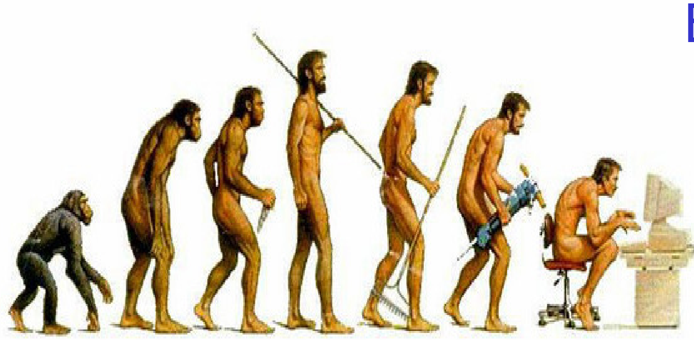
CSE4403 3.0/CSE6602E - Soft Computing
Winter 2011



Lecture 10

Evolutionary & Genetic Programming

Guest lecturer: Xiangdong An
xan@cs.yorku.ca

Evolutionary programming

- By Lawrence J. Fogel in 1960

Sketch of EP

Representation	Real-valued vectors
Parent selection	Deterministic (each parent create one offspring)
Recombination	None
Mutation	Gaussian perturbation
Survivor selection	Probabilistic ($\mu+\mu$)
Speciality	Self-adaptation of mutation step sizes

Representation

- Typically used for continuous parameter optimization
  - A vector of floating-point variables $<x_1, \ldots, x_n>$
  - Objective function: $R^n \rightarrow R$
- To self-adapt mutation parameters
  - $<\underbrace{x_1, \ldots, x_n}_{x}, \underbrace{\sigma_1, \ldots, \sigma_n}_{\sigma}>$

Strategy parameters

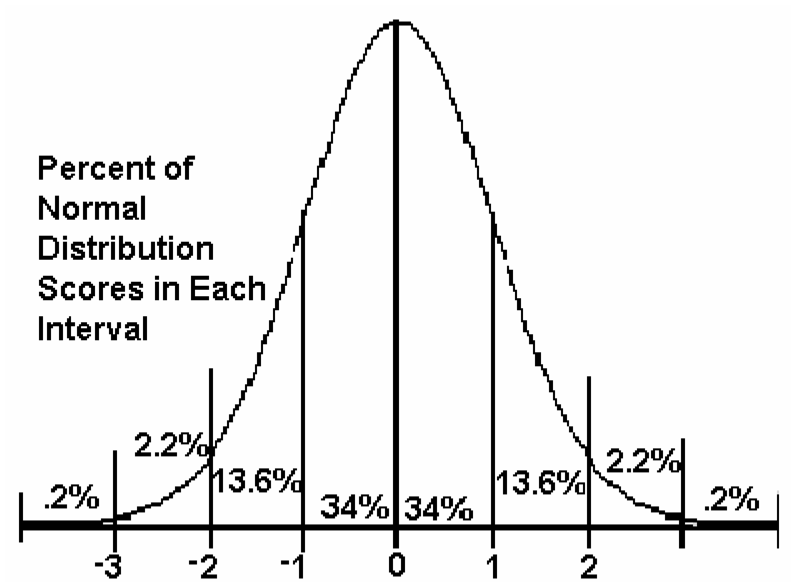
Mutation

$$\sigma_i' = \sigma_i + \alpha * \sigma_i * N(0, 1), \quad \alpha \approx 0.2$$

$$x_i' = x_i + \sigma_i' * N(0, 1)$$

Boundary rule to prevent standard deviation too close to 0:

$$\sigma_i' < \varepsilon_0 \Rightarrow \sigma_i' := \varepsilon_0$$

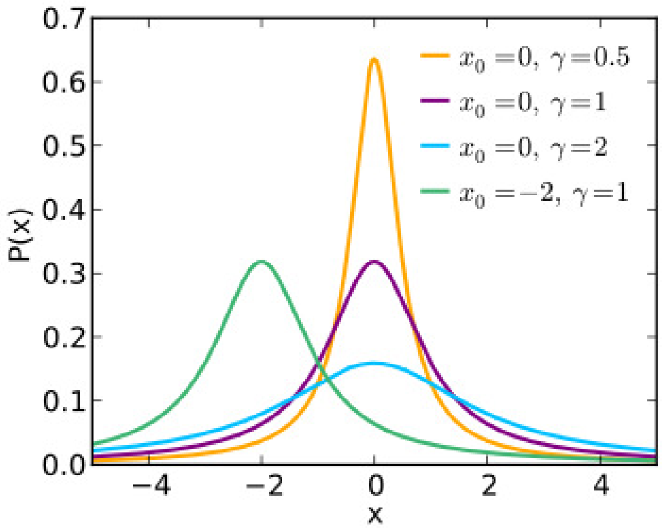


Mutation

- Cauchy distribution can be used to replace normal distribution
- Cauchy distribution has a fatter tail
  - More chance of generating a large mutation and escaping from local minima
  - Gaussian distribution gives greater ability to fine-tune the current parents

$$f(x; x_0, \gamma) = \frac{1}{\pi\gamma \left[1 + \left(\frac{x - x_0}{\gamma}\right)^2\right]}$$

$$= \frac{1}{\pi} \left[\frac{\gamma}{(x - x_0)^2 + \gamma^2}\right]$$



Recombination

- Conceptually not but technically possible
  - It is possible to get improved performance without recombination
  - Depending on the state of the search process(supported by theory)
    - Mutation improves offspring initially
    - Crossover gains in ability as evolution progresses

Parent selection

- Every member creates exactly one offspring by mutation
  - In GA and GP, selective pressure based on fitness
  - In ES, it is stochastic λ/μ

Genetic programming

- Youngest member in EC by Nichael L. Cramer, 1985

Sketch of GP

Representation	Tree structures
Recombination	Exchange of subtrees
Mutation	Random change in trees
Parent selection	Fitness proportional
Survivor selection	Generational replacement

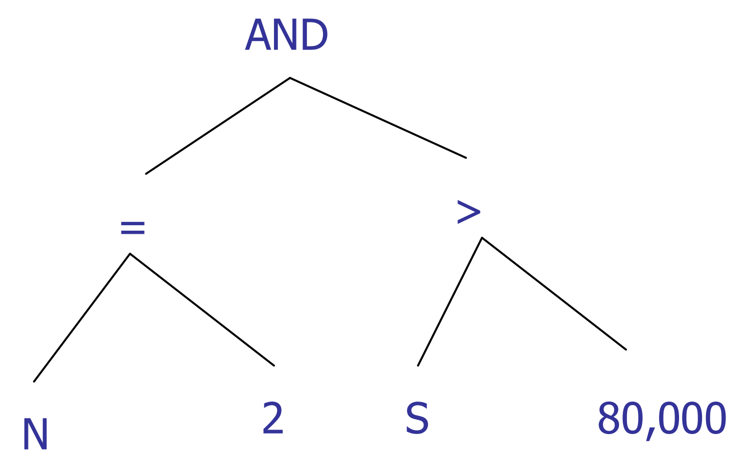
Representation

- Parse trees as chromosomes (non-linear)
  - Function set, e.g. $F = \{+,-,*,/\}$
  - Terminal set, e.g. $T = R \cup \{x, y\}$
  - Rules, e.g.
    - All elements of the terminal set T are correct expressions
    - If $f \in F$ is a function symbol with arity n and $e_1, \ldots, e_n$ are correct expressions, then so is $f(e_1,\ldots,e_n)$.
- Examples
  - Knowledge rules conditions (e.g. classification)
  - Arithmetic expressions
  - Formulas in first-order predicate logic
  - Programming language code

Representation

- Knowledge rules (e.g. classification)

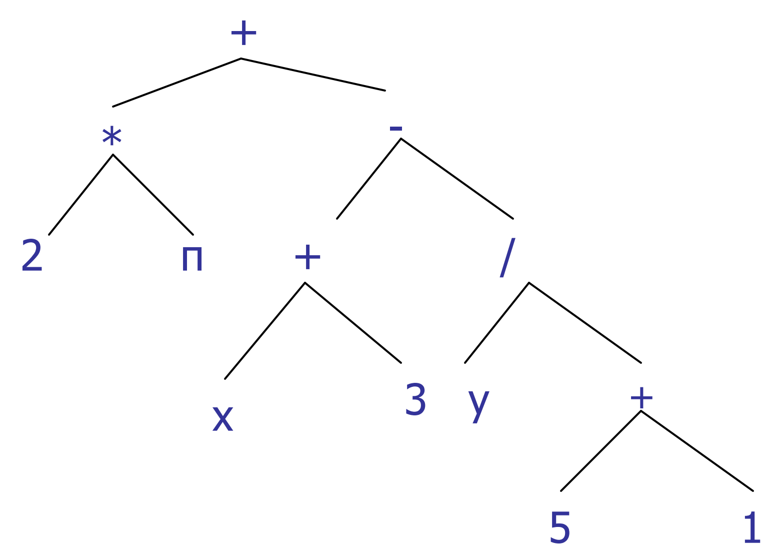
Customer ID	No. of children	Salary	Marital status	Creditworthiness
1	2	45,000	Married	0
2	0	30,000	Single	1
3	1	40,000	Married	1
4	2	60,000	Divorced	1
…	…	…	…	…
10000	2	50,000	Married	1



IF (**N**o. children = 2) AND (**S**alary > 80,000) THEN good ELSE bad

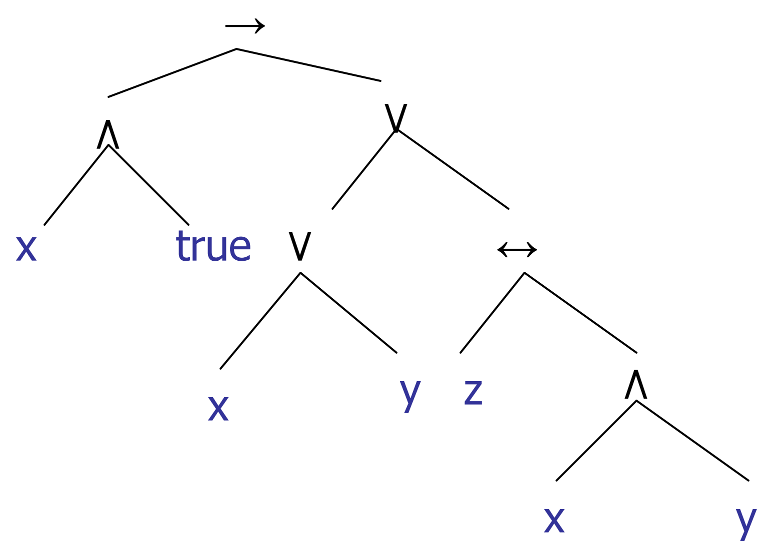
Representation

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



Representation

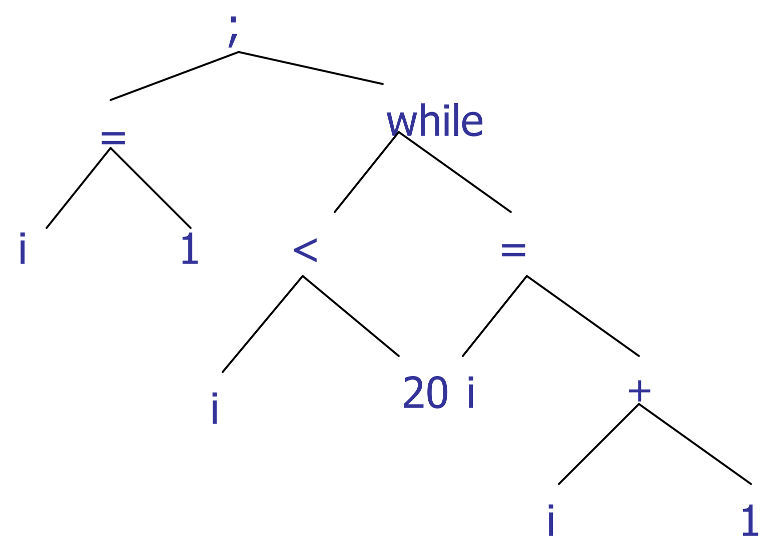
- Formulas in first-order predicate logic
  - $(x \wedge true) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$



Representation

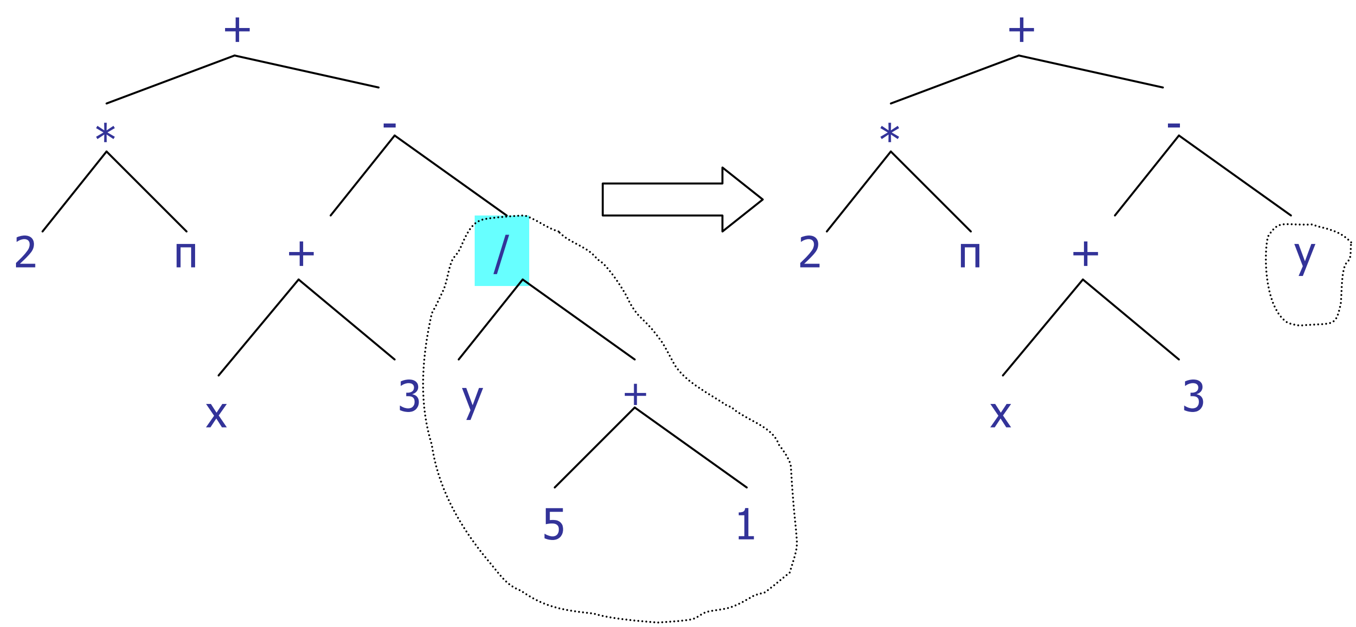
- Programming language code

```
i = 1;
while (i < 20){
    i=i+1;
}
```



Mutation

- GP is a variant of GA with a different data structure - tree
  - Replacing a random subtree by a randomly generated tree
    - The probability of mutation at the junction with recombination
    - The probability of choosing a subtree to be replaced

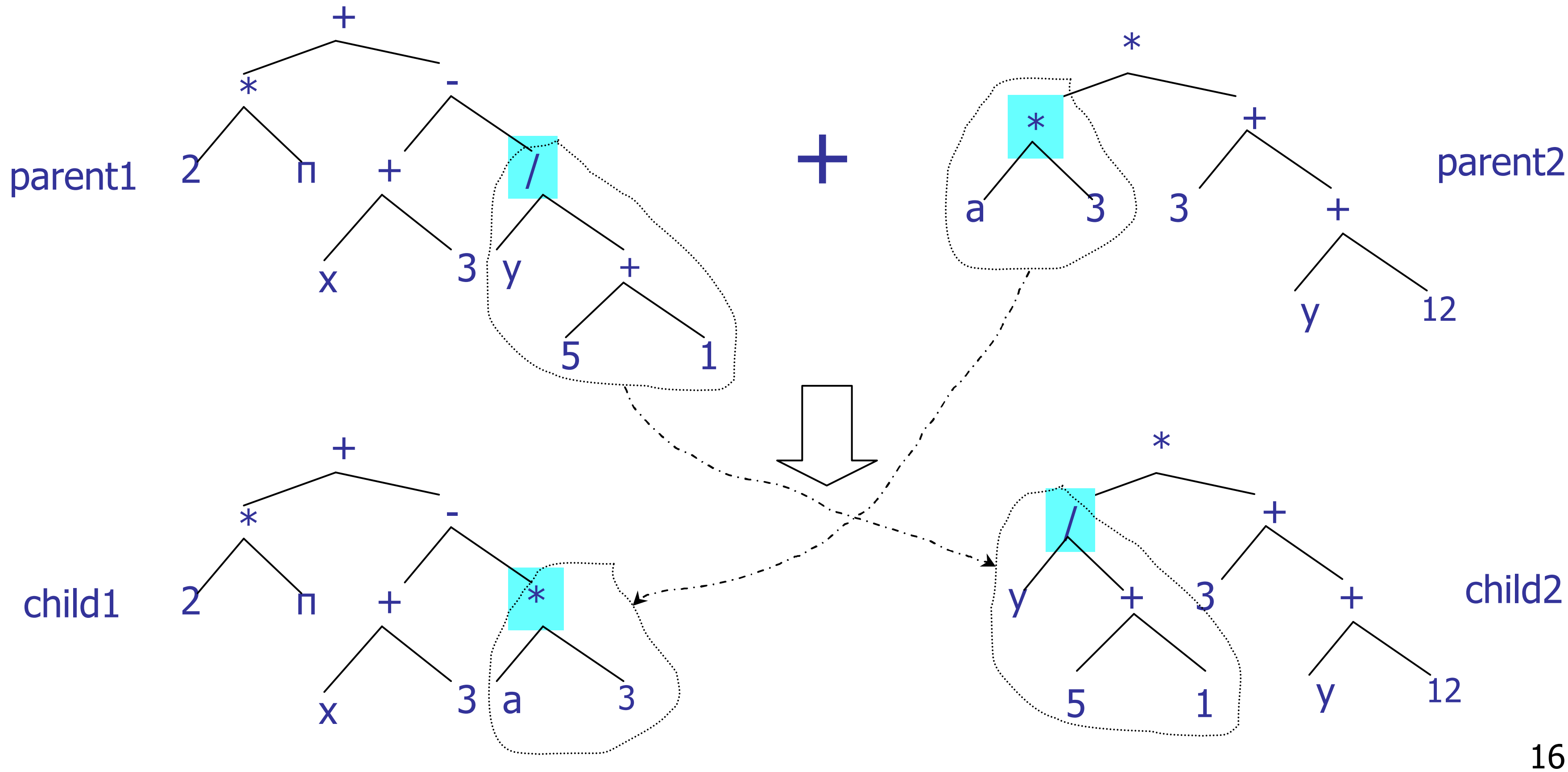


Mutation

- It is suggested GP works without mutation or by 5%
  - This makes GP different from other EAs
  - The crossover has a large shuffling effect, accounting in some sense as a macromutation operator

Recombination

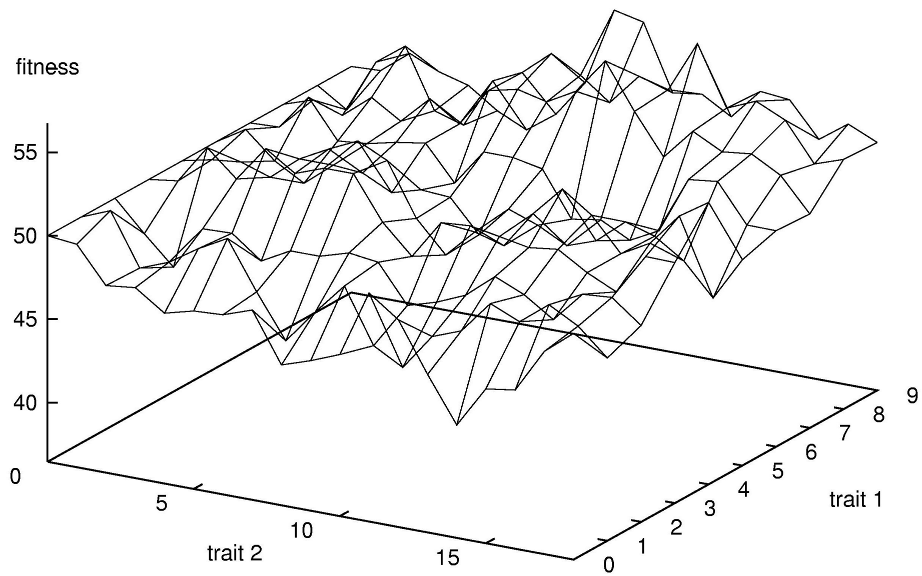
- Swapping subtrees among selected parents (subtree crossover)



Discussion on evolutionary computing

- Unimodal problems
  - A problem has only one point that is fitter than all points
  - Global optimum
- Multimodal problems
  - A problem has multiple points that are fitter than their neighbors
  - Local optima and global optimum
- Genetic drift
  - Variety or highly fit individuals are lost from the population
  - Climbs the wrong hill - local optima

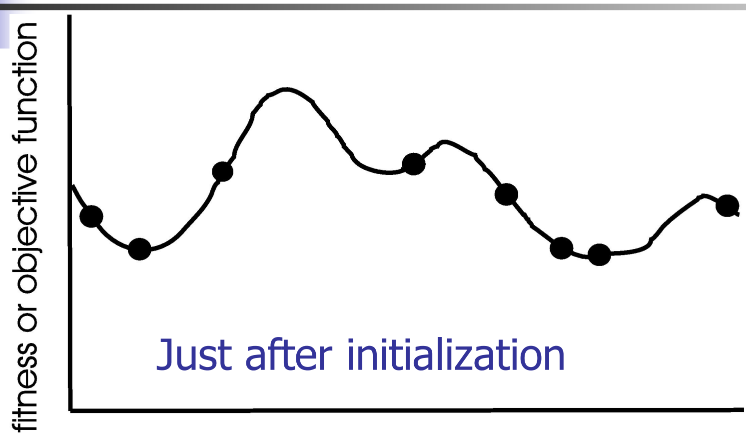
Adaptive landscape (surface)



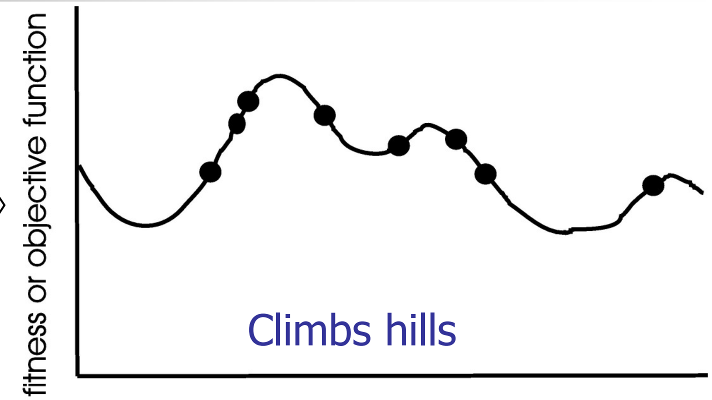
Discussion on evolutionary computing

- Exploration
  - The generation of new individuals in as yet untested regions
  - If too much, it is inefficient
- Exploitation
  - The concentration of the search in the vicinity of current individuals
  - If too much, premature convergence
    - Losing population diversity too quickly & get trapped in a local optimum
- Trade-off between exploration and exploitation

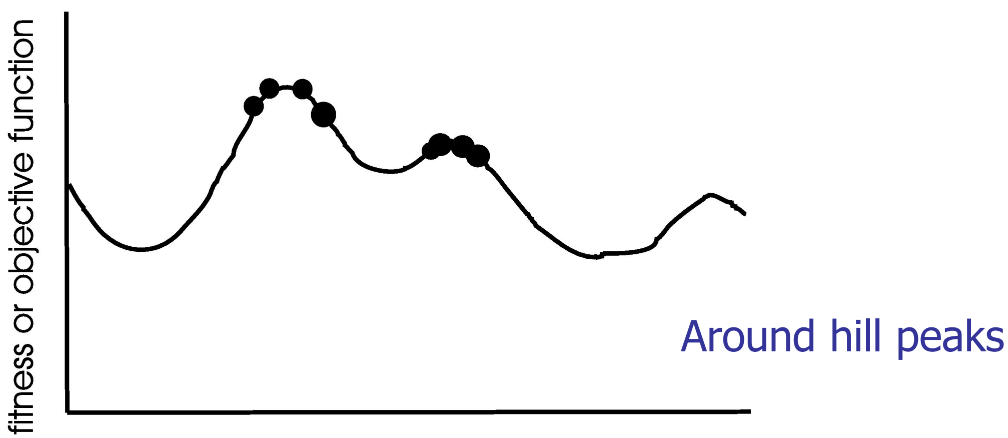
Working of evolutionary algorithms



- individuals or points of obj. function domain



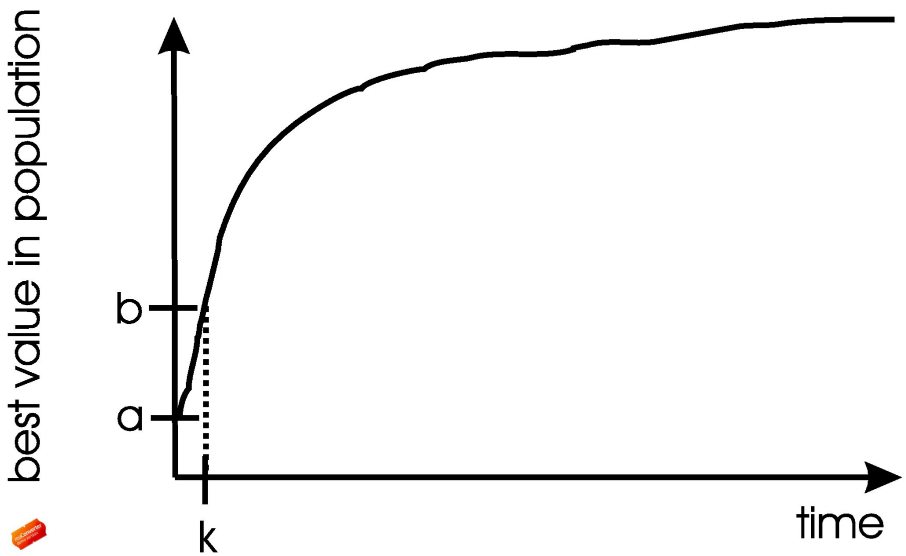
- individuals or points of obj. function domain



- individuals or points of obj. function domain

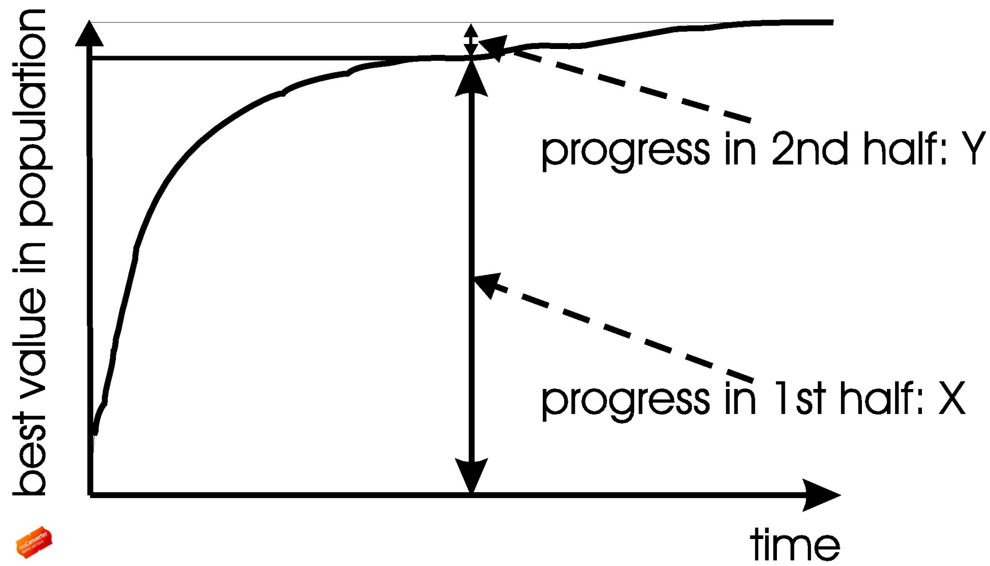
A typical 3 step process

Better initialization?



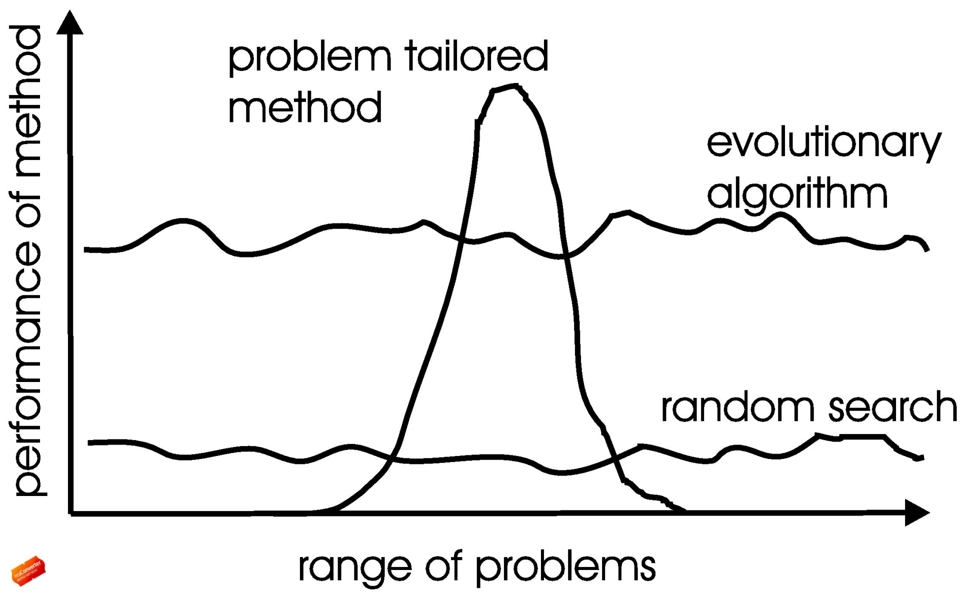
After a small *k* generations, population quality can reach level *b* from level *a.* The worth of extra effort to start from a better population is questionable.

Long runs?



It might not be worthwhile to allow very long runs

Better than random search



No problem or instance-specific knowledge incorporated

However, the No Free Lunch theorem has shown no blackbox algorithm can outperfrom random walk when averaged over “all” problems.