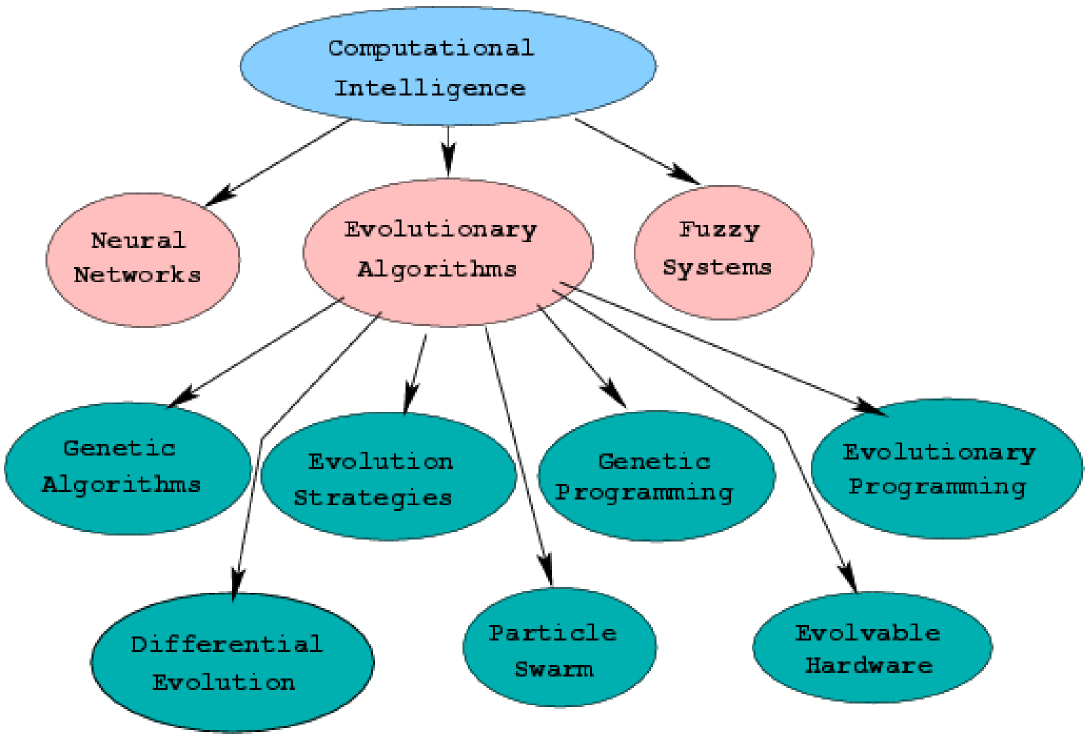
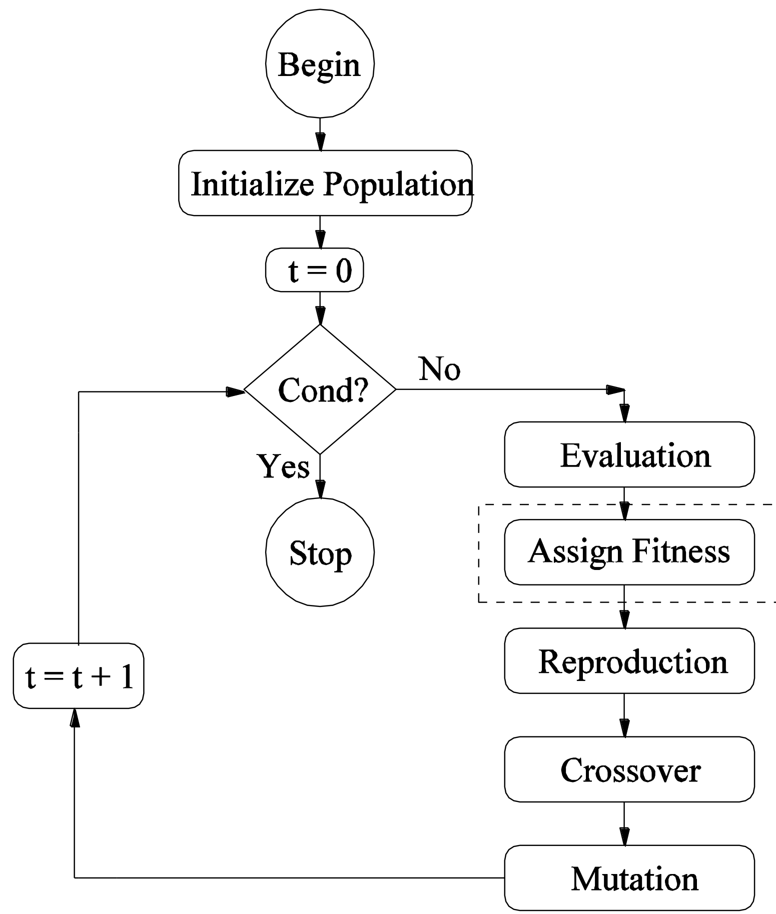
Computational Intelligence and Evolutionary Algorithms (EAs)



- We treat an EA here as a search and optimization tool

Evolutionary Algorithms as Optimizers

```
begin
t := 0;
Initialize P(t);
Evaluate P(t);
while not Terminate
do
   P'(t)   := Selection (P(t));
   P''(t)  := Variation (P'(t));
   Evaluate P''(t);
   P(t+1):= Survivor (P(t),P''(t));
    t := t+1;
od
end
```

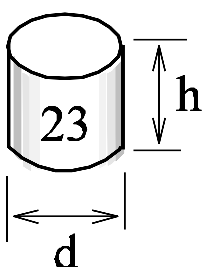


Evolutionary Algorithm Operators

- Initialization of a set of candidate solutions: *Population*
- Create new solutions by:
  - Reproduction: Copy good individuals (Survival-of-the-fittest principle)
  - Recombination or Crossover: ≥ 2 parents $\rightarrow \geq 1$ offspring
  - Mutation: 1 parent $\rightarrow$ 1 offspring
- Evaluation of solution: Objective function $\rightarrow$ Fitness
- Uses an elite-preservation principle

Binary-Coded Genetic Algorithms

- Genetic Algorithms (John Holland, 1962)
- Design of a can for minimum cost having at least ***V*** volume
- Objective function: Cost $f(d,h) = \pi dh + 2\pi d^2/4$
- Constraint: Volume $\pi d^2 h/4 \geq 400$
- Representation in binary strings:
- Fitness: objective value + penalty for constraint



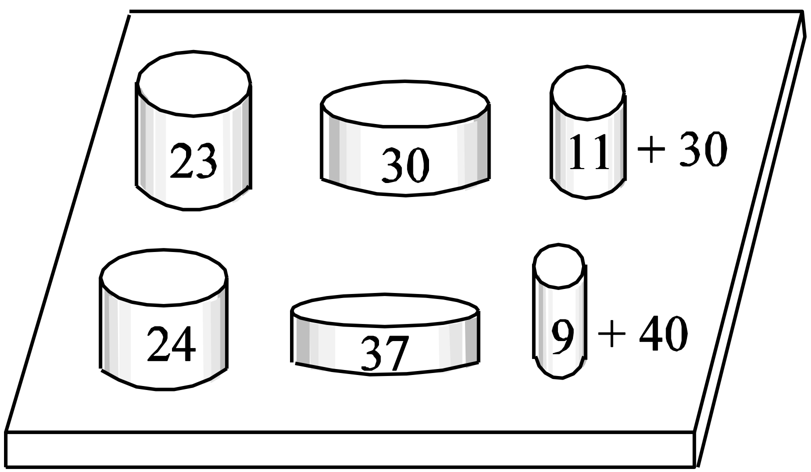
(d, h) = (8, 10) cm

(Chromosome) = 0 1 0 0 0 0 1 0 1 0

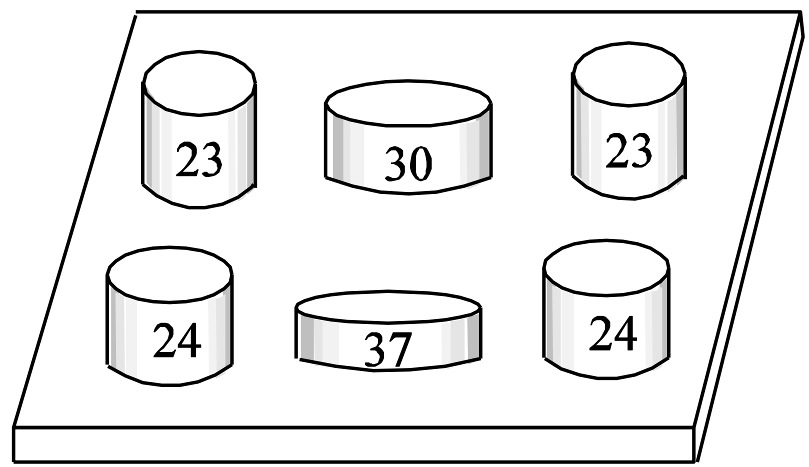
Genetic Algorithm: A Hand Simulation

- Fitness = Cost + Penalty (proportional to constraint violation)

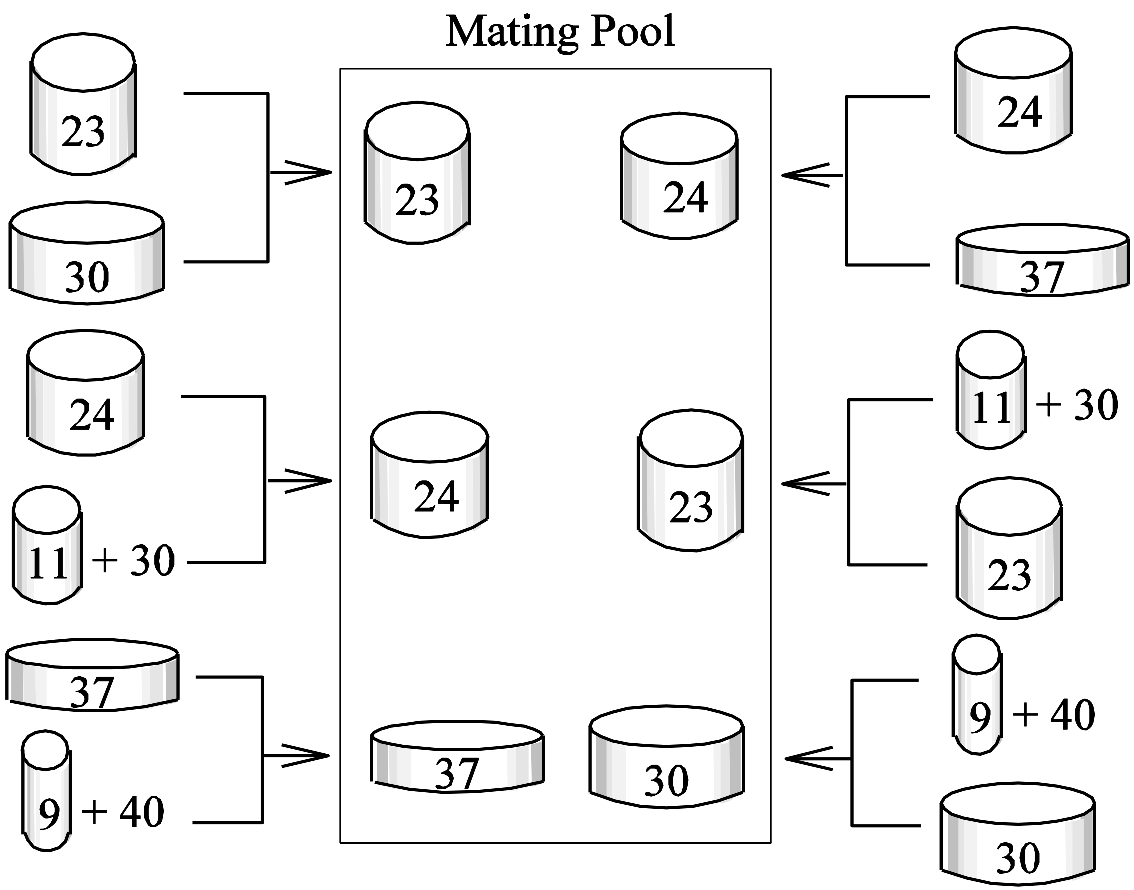
Random Initialization



Population after Selection

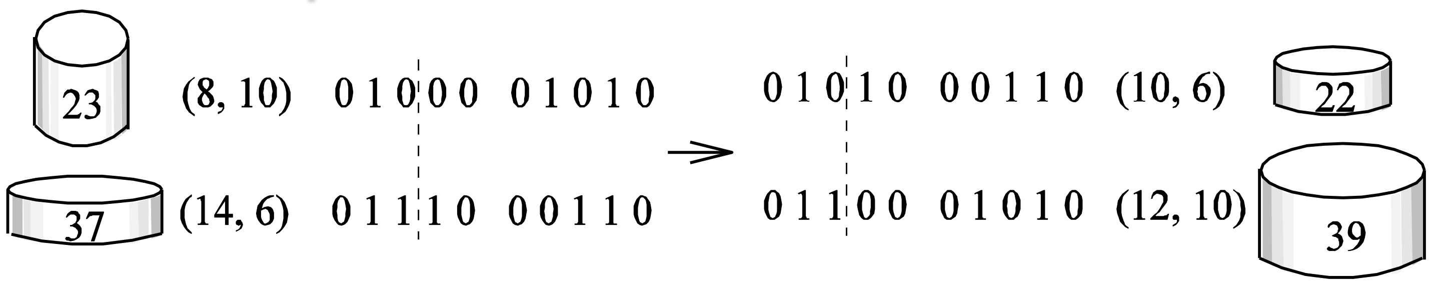


Tournament Selection Operator

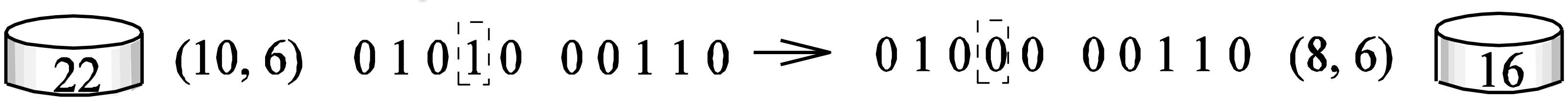


Variation Operators

- Crossover operator:



- Mutation operator:



- Good, partial information propagates leading to optimum
- Other and modified operators often used

Advantages of EAs

- Applicable in problems where no (good) method is available
  - Discontinuities, non-linear constraints, multi-modalities
  - Discrete variable space
  - Implicitly defined models (*if-then-else*)
  - Noisy/dynamic problems
- Most suitable in problems where multiple solutions are sought
  - Multi-modal optimization problems
  - Multi-objective optimization problems
- Parallel implementation easier

Disadvantages of EAs

- No guarantee for finding optimal solutions in a finite amount of time
  - However, asymptotic convergence proofs are available
  - For specific problems, computational complexity worked out
- Parameter tuning mostly by trial-and-error:

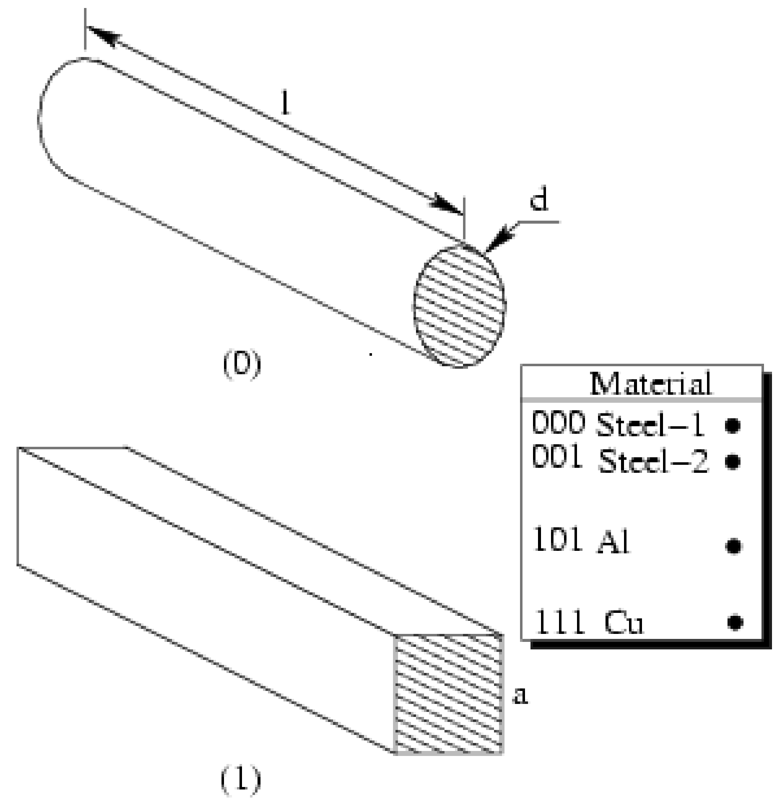
  Self-adaptation
- Population approach may be expensive: Parallelism

EAs in Engineering Optimization

- Handling mixed variables: Boolean, discrete, real etc.
- Handling non-linear constraints
- Handling large-scale problems
- Handling multi-modal problems
- Handling multiple conflicting objectives
- Handling computationally-expensive problems
- Handling uncertainties

Handling Mixed Variables

- Level-wise application in classical methods
- No need for level-wise optimization with EAs
- A mixed representation possible:
  (1) 14 23.457 (101)
- Recombination and mutation can be used variable-wise
- How to handle real-parameters in EAs?



Real-Parameter Evolutionary Algorithms

- Decision variables are coded directly, instead of using binary strings

- Recombination and mutation need structural changes

Recombination

$$\begin{matrix}(x_1 x_2 \ldots\ldots\ldots\ldots x_n) \\ (y_1 y_2 \ldots\ldots\ldots\ldots y_n)\end{matrix} \Rightarrow ?$$

Mutation

$$(x_1 x_2 \ldots\ldots\ldots\ldots\ldots x_n) \Rightarrow ?$$

- Simple exchanges are not adequate

Different Real-Parameter Evolutionary Algorithms

- Evolution strategy (ES):
  - Correlated self-adaptive evolution strategies
  - Covariance Matrix Adaptation (CMA)
- Differential evolution
- Particle swarm optimization (PSO)
- Real-parameter genetic algorithms
  - BLX, UNDX, SBX, SPX, arithmetic crossover, Gaussian mutation etc.

Simulated Binary Crossover (SBX)

- Step 1: Choose a random number

$$u \in [0,1].$$

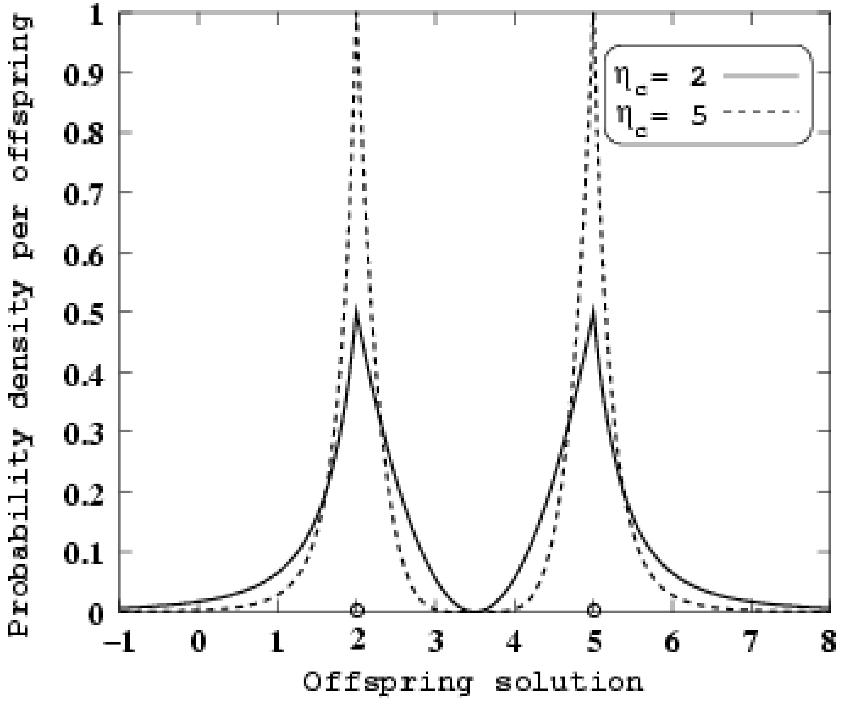
- Step 2: Calculate β_q:

$$\beta_q = \begin{cases} (2u)^{\frac{1}{\eta_c+1}}, & \text{if } u \le 0.5 \\ \left(\dfrac{1}{2(1-u)}\right)^{\frac{1}{\eta_c+1}}, & \text{otherwise} \end{cases}$$

- Step 3: Compute two offspring:

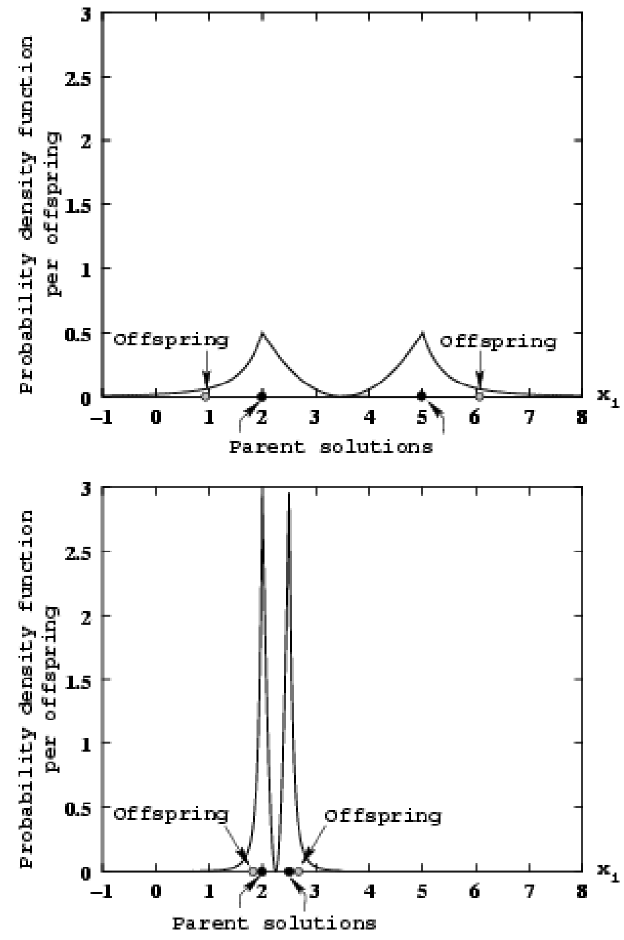
$$c_1 = 0.5\left(\left(1+\beta_q\right)p_1 + \left(1-\beta_q\right)p_2\right)$$

$$c_2 = 0.5\left(\left(1-\beta_q\right)p_1 + \left(1+\beta_q\right)p_2\right)$$

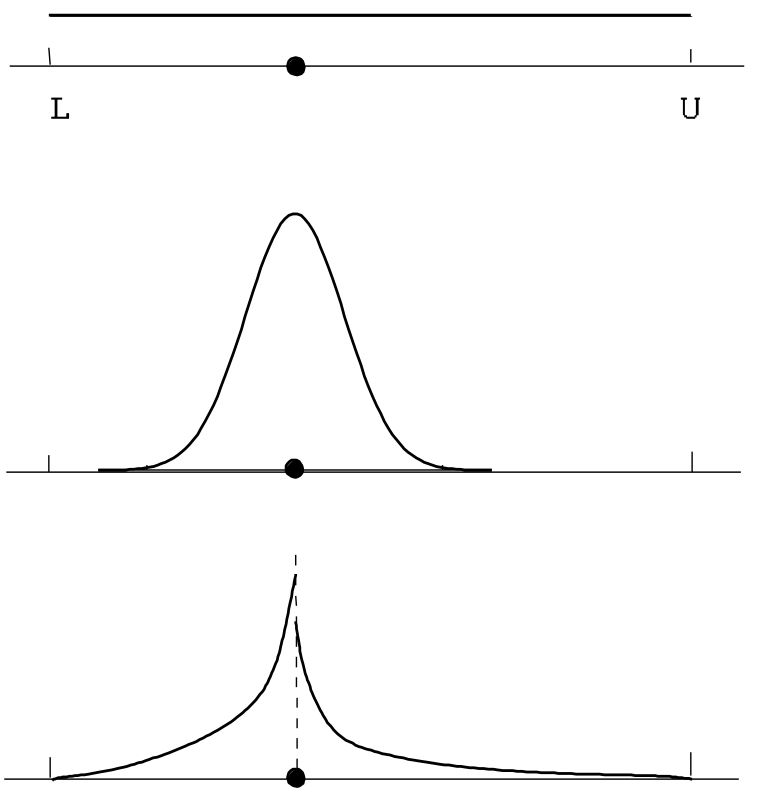


Properties of SBX Operator

- If parents are distant, distant offspring are likely
- If parents are close, offspring are close to parents
- Self-adaptive property



Mutation Operators



- Random mutation
- Normally distributed mutation
- Non-uniform mutation