CO3091 - Computational Intelligence and Software Engineering

Lecture 05

Evolutionary Algorithms — Part II

Leandro L. Minku

Overview — Previous Lecture

Some concepts from natural evolution.

Evolutionary algorithms.

- Representation.
- Initialisation of the population.
- Determining the fitness of individuals.

Overview

Evolutionary Algorithm

1. Initialise population
2. Evaluate each individual (determine their fitness)
3. Repeat (until a termination condition is satisfied)

    3.1 Select parents

    3.2 Recombine parents with probability Pc

    3.3 Mutate resulting offspring with probability Pm

    3.4 Evaluate offspring

    3.5 Select survivors for the next generation

Overview — Today's Lecture

Evolutionary Algorithm

1. Initialise population
2. Evaluate each individual (determine their fitness)
3. Repeat (until a termination condition is satisfied)

    3.1 Select parents

    3.2 Recombine parents with probability Pc

    3.3 Mutate resulting offspring with probability Pm

    3.4 Evaluate offspring

    3.5 Select survivors for the next generation

EA's Pseudocode

Evolutionary Algorithm

1. Initialise population
2. Evaluate each individual (determine their fitness)
3. Repeat (until a termination condition is satisfied)

    3.1 Select parents

    3.2 Recombine parents with probability Pc

    3.3 Mutate resulting offspring with probability Pm

    3.4 Evaluate offspring

    3.5 Select survivors for the next generation

Parent Selection

- Usually probabilistic:
  - Better fit solutions more likely to become parents than less fit solutions.
  - Even the worst in current population usually has non-zero probability of becoming a parent.

- This stochastic nature can help to escape from local optima.

Image from: https://lh6.googleusercontent.com/-wCEtlOfs4Il/TXjes2fSfaI/AAAAAAAABEg/7yOX_b1D2Ho/s1600/pavoesMenor.jpg

Parent Selection Mechanisms

- Roulette Wheel
- Tournament Selection
- Ranking Selection

Parent Selection Mechanisms

- **Roulette Wheel**
- **Tournament Selection**
- Ranking Selection

Roulette Wheel Parents Selection

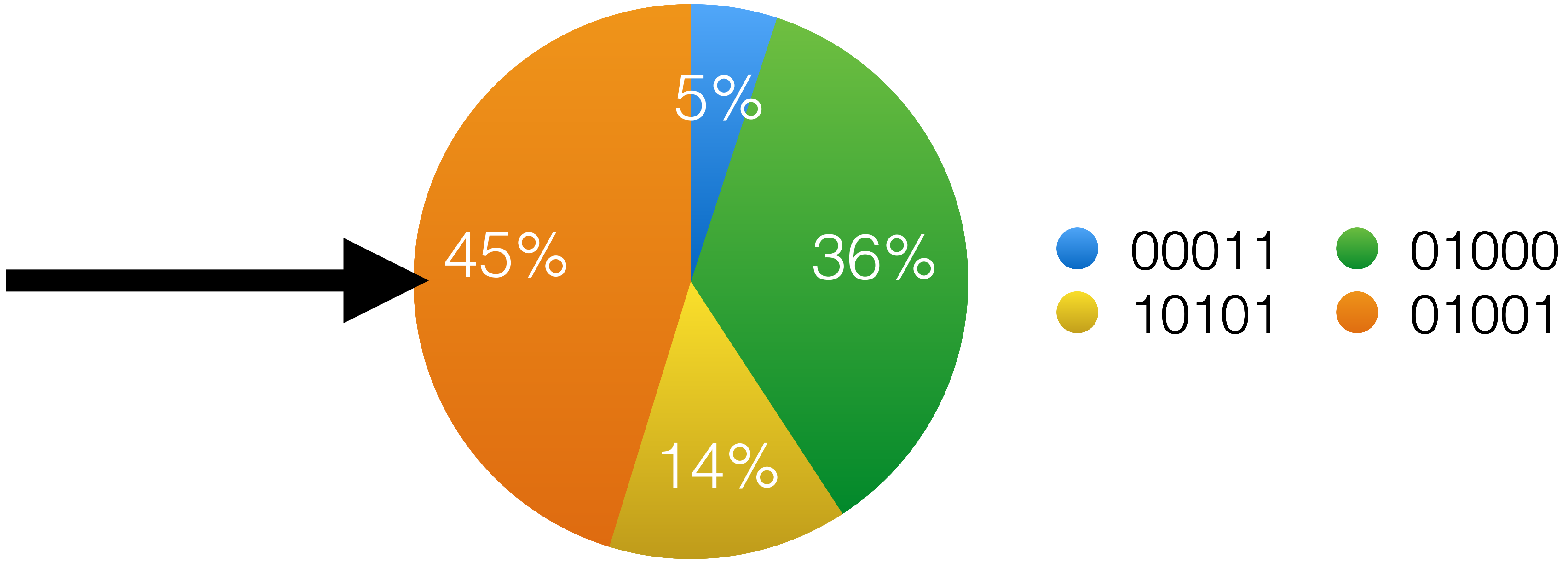
- Probability of an individual to be selected as parent is proportional to its fitness. Assuming maximisation of positive fitnesses: $f(x) / \Sigma f(x)$.

- Example:
  - Problem: maximise $f(x) = x^2$, $x \in \{-15,-14,\ldots,0,1,2,\ldots,15\}$
  - Representation: $\{0,1\}^5$.

Genotypes	Phenotypes	Fitnesses	Probability
00011	3	9	9/179 = 0.0503
01000	8	64	64/179 = 0.3575
10101	-5	25	25/179 = 0.1397
01001	9	81	81/179 = 0.4525
Sum (Σ):		179	1

Roulette Wheel Parents Selection — Selecting 4 Parents

Genotypes	Phenotypes	Fitnesses	Probability
00011	3	9	9/179 = 0.0503
01000	8	64	64/179 = 0.3575
10101	-5	25	25/179 = 0.1397
01001	9	81	81/179 = 0.4525
Sum (Σ):		179	1



Randomly selected Parents:

01001
10101
01000
01000

Problems of Roulette Wheel Parents Selection

- Outstanding individuals may take over the population very quickly, causing premature convergence.
- When fitness values are very close to each other, there is almost no selection pressure.
- The mechanism behaves differently on transposed versions of the same function.

Problems of Roulette Wheel Parents Selection

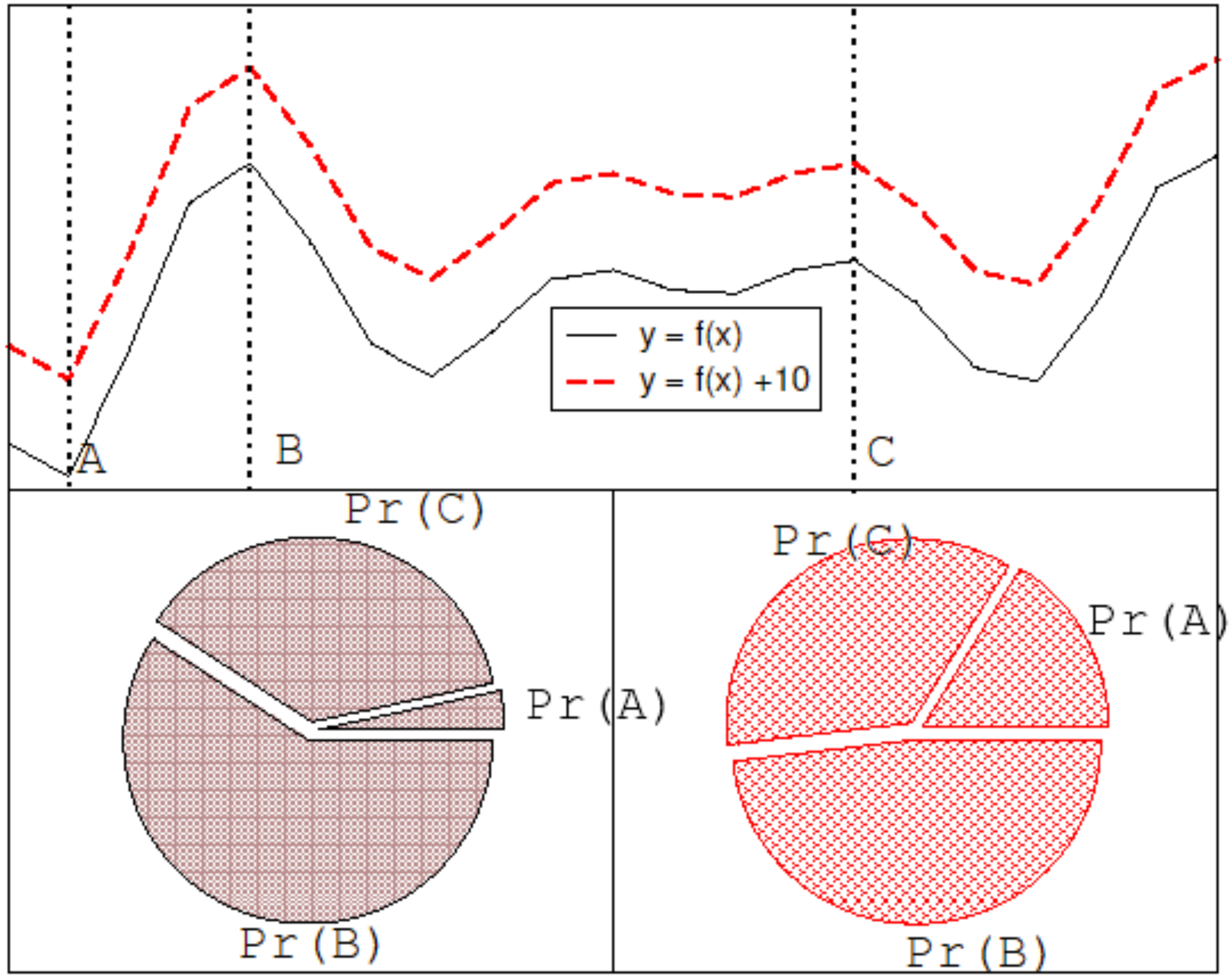


Image from Eiben and Smith's slides.

Tournament Selection

- Informal Procedure:
  - Pick *k* individuals at random then select the best of these.
  - Repeat to select more individuals.

E.g.: k = 2, assuming maximisation

Genotypes	Phenotypes	Fitnesses
→ 00011	3	9
01000	8	64
10101	-5	25
→ 01001	9	81

Parent: 01001

How Many Parents to Select?

- This is a design choice of the algorithm.
- Frequently, if your population size is *S*, you choose the number of parents so as to produce *S* children.
- E.g., if each pair of parents can produce 2 children by recombination, you could select *S* parents to produce *S* children.

EA's Pseudocode

Evolutionary Algorithm

1. Initialise population
2. Evaluate each individual (determine their fitness)
3. Repeat (until a termination condition is satisfied)

    3.1 Select parents

    3.2 Recombine parents with probability Pc

    3.3 Mutate resulting offspring with probability Pm

    3.4 Evaluate offspring

    3.5 Select survivors for the next generation

Recombination

- Creates offspring based on parent individuals.
- We have a certain probability that recombination between parents will occur. If it doesn't occur, we clone the parents.
- Most offspring may be worse or similar to the parents, because the choice of what genes from what parent goes to the offspring is random.
- Hope: some offspring are better, by combining the good elements of the genotype of each parent.
- Principle used for millennia by breeders for plants and livestock.
- Genetic algorithms are evolutionary algorithms that give high emphasis to recombination (probability is typically in [0.6,0.9]).

Recombination Operators for Binary, Integer and Floating Point Vectors

- 1-Point Crossover
- Multi-parent recombination
- Uniform Crossover
- N-Point Crossover
- …

PS: the term crossover is usually used for recombination involving 2 parents.

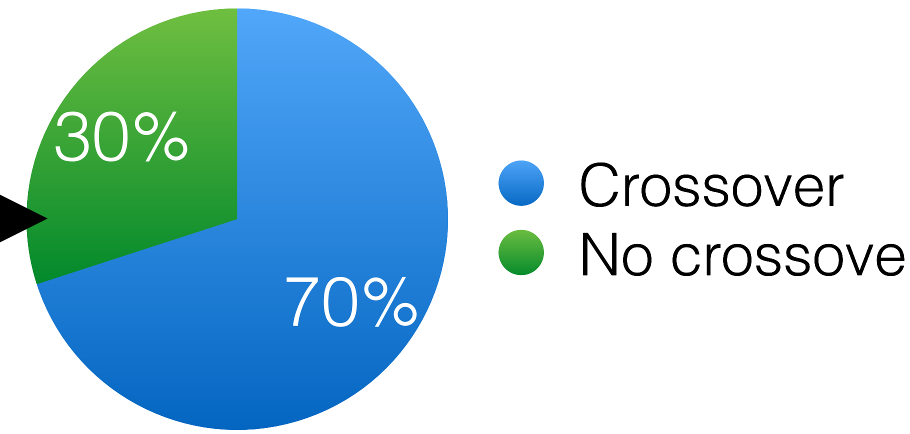
Recombination Operators for Binary, Integer and Floating Point Vectors

- **1-Point Crossover**
- **Multi-parent recombination**
- Uniform Crossover
- N-Point Crossover
- …

PS: the term crossover is usually used for recombination involving 2 parents.

1-Point Crossover

- Select a random point.
- Split parents at this point.
- Exchange tails to create children.
- Example:



Parents	0 1 | 0 0 1	0 1 0 | 0 0	0 1 0 0 1
	1 0 | 1 0 1	0 1 0 | 0 0	1 0 1 0 1
Offspring	0 1 1 0 1	0 1 0 0 0	0 1 0 0 1
	1 0 0 0 1	0 1 0 0 0	1 0 1 0 1

1-Point Crossover

- 1-point crossover:

Parents 0 1 | 0 0 1
1 0 | 1 0 1

Offspring 0 1 1 0 1
1 0 0 0 1

- 1-point crossover can also be used for integer and floating point vectors.

Problem of 1-Point Crossover

- Positional bias:
  - Performance depends on the order that components of the design variable occur in the representation.
  - More likely to keep together genes that are near each other.
  - Can never keep together genes from opposite ends of vector.
  - Can be exploited if we know about the structure of our problem, but this is not usually the case.

Multi-Parent Recombination

- Multi-parent recombination.
  - E.g.:

Image from Channel 4 animation.

[Youtube video posted by CCTV News (Feb 2015): https://youtu.be/GcubrH6HRnk]

Diagonal "Crossover"

- For *k* parents, select *k-1* crossover points.
- Create *k* children diagonally.
- Alternatively, create only the first of the children.

Parents

1 0 | 0 1 0 | 0 0 1
0 1 | 0 1 1 | 0 1 1
0 0 | 1 1 1 | 0 1 0

Offspring

1 0 0 1 1 0 1 0
0 1 1 1 1 0 0 1
0 0 0 1 0 0 1 1

Other Recombination Operators for Floating-Point Representation

- Intermediate recombination
  - Simple average between parents

10.2, 5.6, 3.4
5.2, 4.4, 10.4
→ 7.7, 5.0, 6.9

EA's Pseudocode

Evolutionary Algorithm

1. Initialise population
2. Evaluate each individual (determine their fitness)
3. Repeat (until a termination condition is satisfied)

    3.1 Select parents

    3.2 Recombine parents with probability Pc

    3.3 Mutate resulting offspring with probability Pm

    3.4 Evaluate offspring

    3.5 Select survivors for the next generation

Mutation

- Acts on one genotype and generates another.
- Typically causes small changes.
- Randomness differentiates it from other unary heuristic operators.
- Can introduce traits that were originally inexistent in a population.

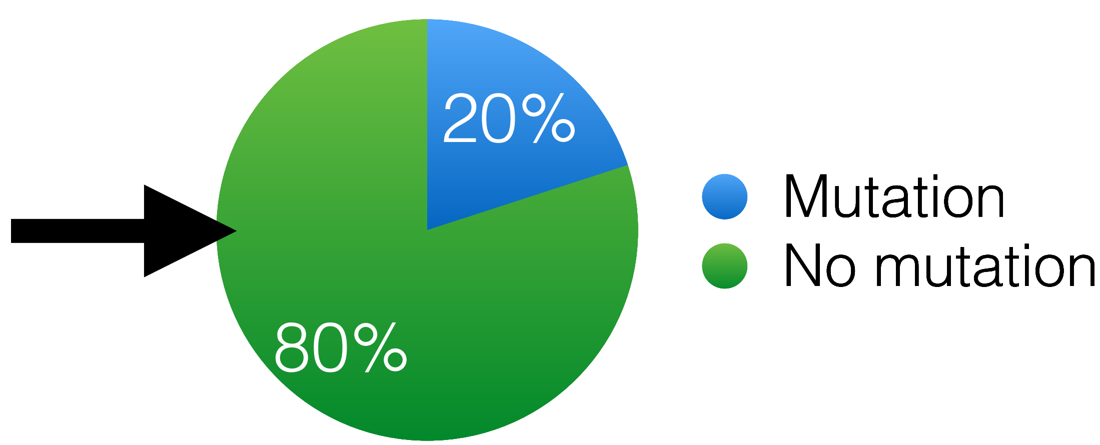
Image from: http://www.cheatsheet.com/wp-content/uploads/2014/01/Waterworld.jpg?

Bitwise Bit-Flipping Mutation

- Flip each gene (from 0 to 1 or vice-versa) with probability *Pm*.
- *Pm* is called mutation rate, and is typically in [1/pop_size, 1/chromosome_length].
- Example:

Before mutation 0 1 1 0 1

After mutation 0 1 1 **1** 1



Mutation for Integer Representation

- Random Reset
- Creep Mutation

Random Reset

- More adequate for categorical design variables.
  - E.g., design variable in {Toyota, Volkswagen, Fiat, Vauxhall, BMW, Mercedes}.

- Pick a new value from the permissible set of values.

- Same probability for all possible values (discrete uniform distribution).

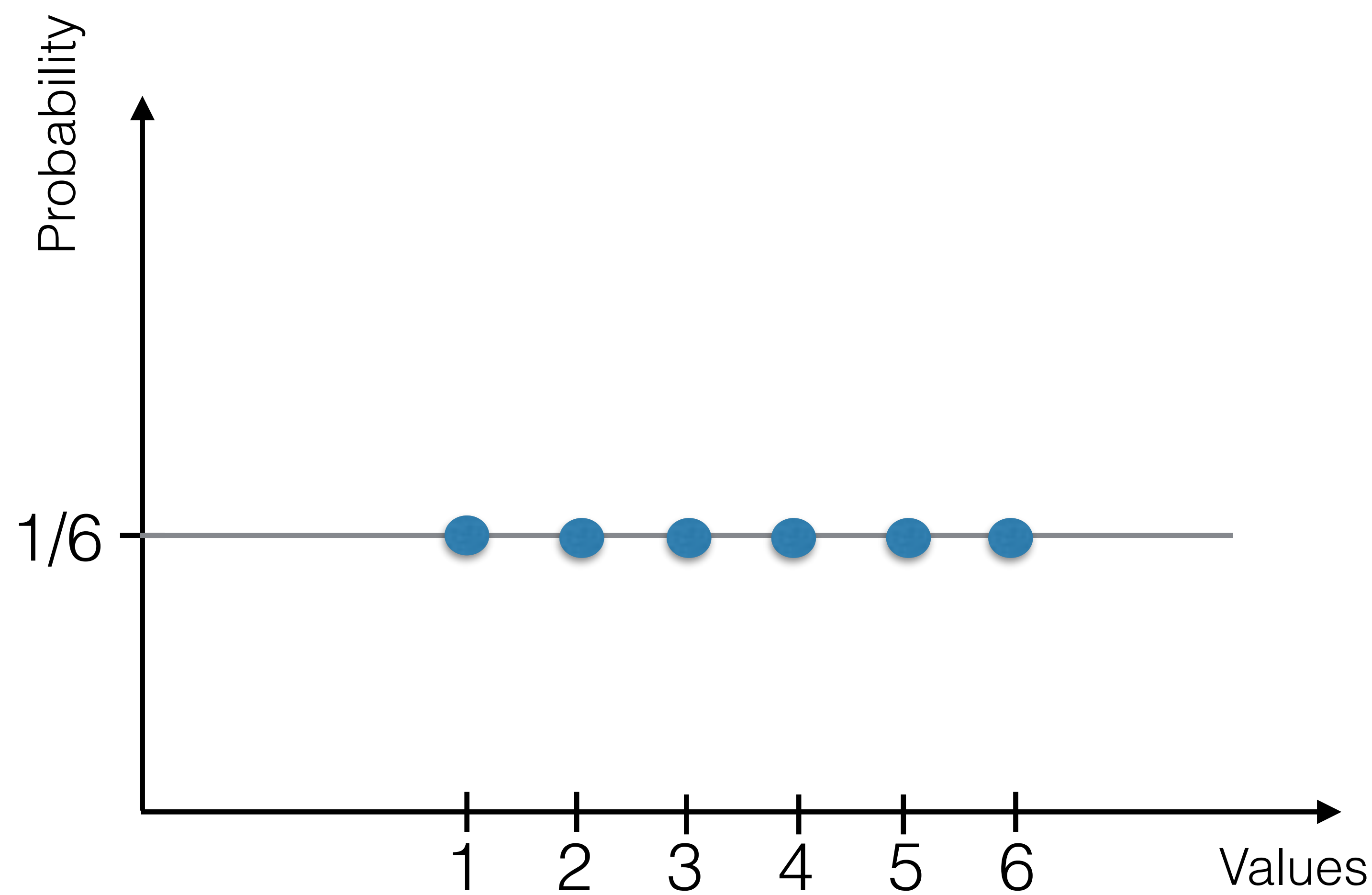
- E.g., let's say that your design variable could take any value in {1,2,…, 6} in your integer representation.

$$1 \longrightarrow 2$$

equally likely to

$$1 \longrightarrow 5$$

Discrete Uniform Distribution



Creep Mutation

- More adequate for ordinal design variables.
  - E.g., for the problem of maximising $f(x) = x^2$, $x \in \{1,2,\ldots,6\}$.

- Change the gene value to another value in the following way:
  - Small changes should be more likely than bigger changes.

$1 \longrightarrow 2$ more likely

$1 \longrightarrow 5$ less likely

  - Increases or reductions are equally likely.

$5 \longrightarrow 6$

equally likely

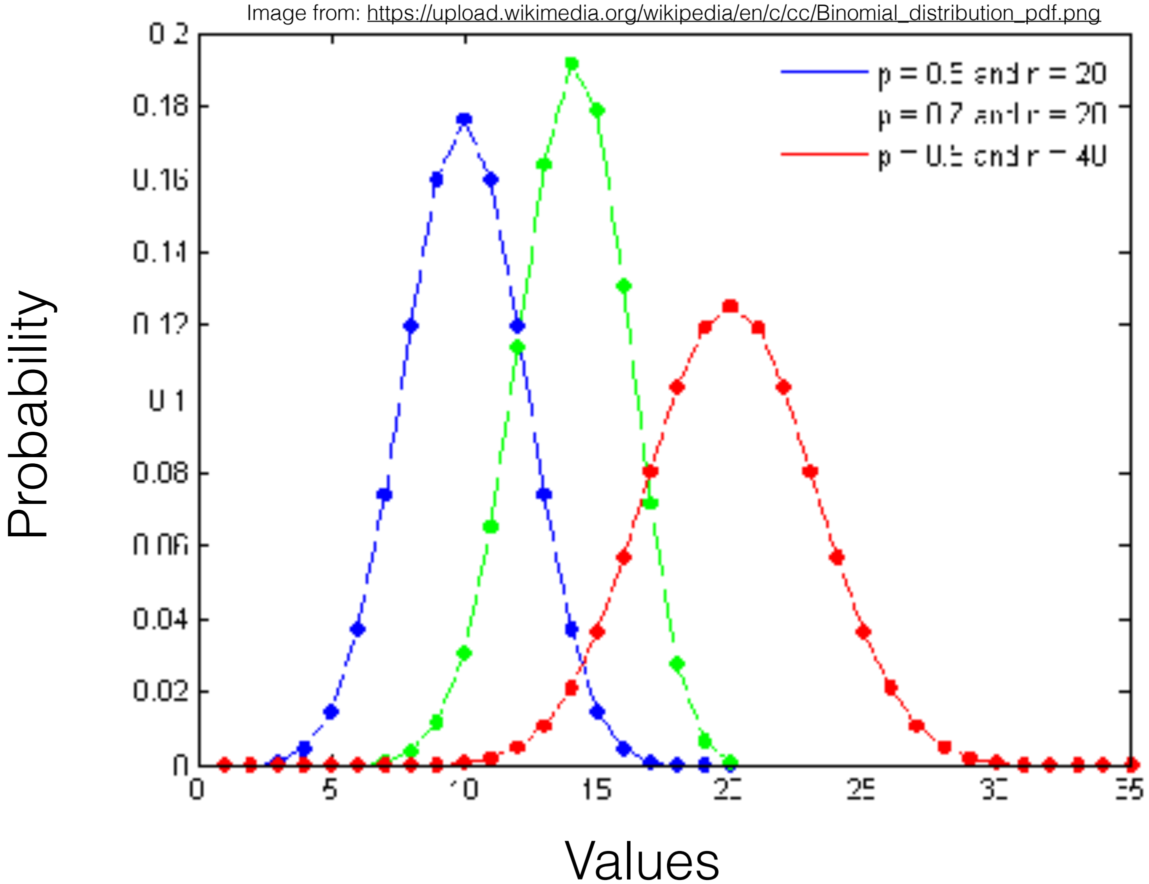
$5 \longrightarrow 4$

  - Pick the new value from a symmetric probability distribution centred at the current value.

Creep Mutation

Binomial Distribution B(n,p)

Image from: https://upload.wikimedia.org/wikipedia/en/c/cc/Binomial_distribution_pdf.png



E.g., pick a value from a binomial distribution, and subtract a certain amount (the mean of the distribution) to centre it at zero.

Mutation for Floating Point Representation

- Uniform
- Non-Uniform

Uniform Mutation for Floating Point Representation

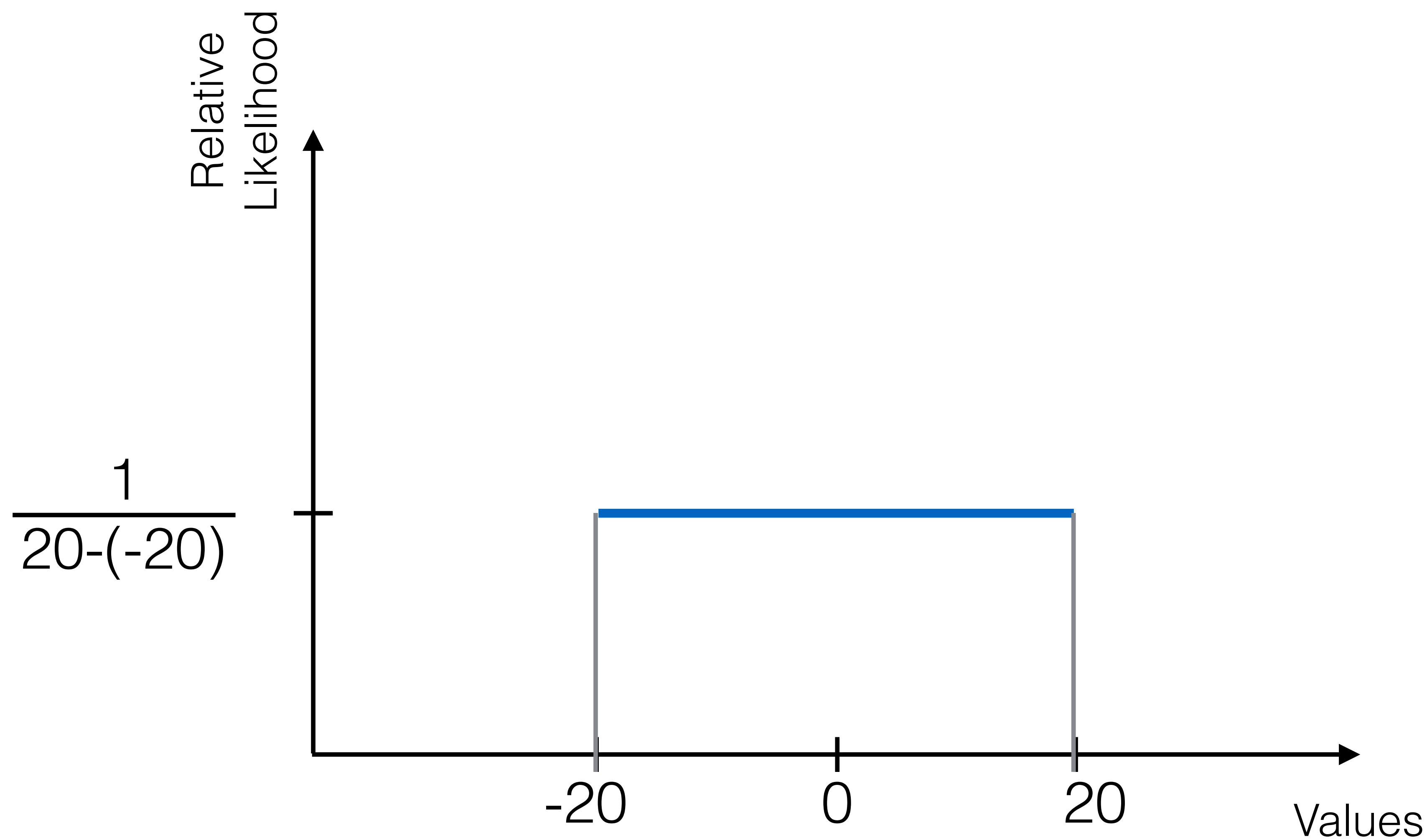
- Similar to random resetting.
- Take a new number among all possible numbers.
- Same probability for all numbers (uniform distribution).
- E.g., let’s say your design variable can take any value in [-20,20] in your representation.

$5.2 \longrightarrow 5.5$

equally likely to

$5.2 \longrightarrow 15.7$

Continuous Uniform Distribution



Non-Uniform Mutation for Floating Point Representation

- Similar to Creep Mutation.

- E.g.: Change the floating point value to another one in the following way:
  - Small changes should be more likely than bigger changes.

    $5.2 \longrightarrow 5.5$ more likely

    $5.2 \longrightarrow 15.7$ less likely

  - Increases and reductions are equally likely.

    $5.2 \longrightarrow 5.5$

    $5.2 \longrightarrow 4.9$

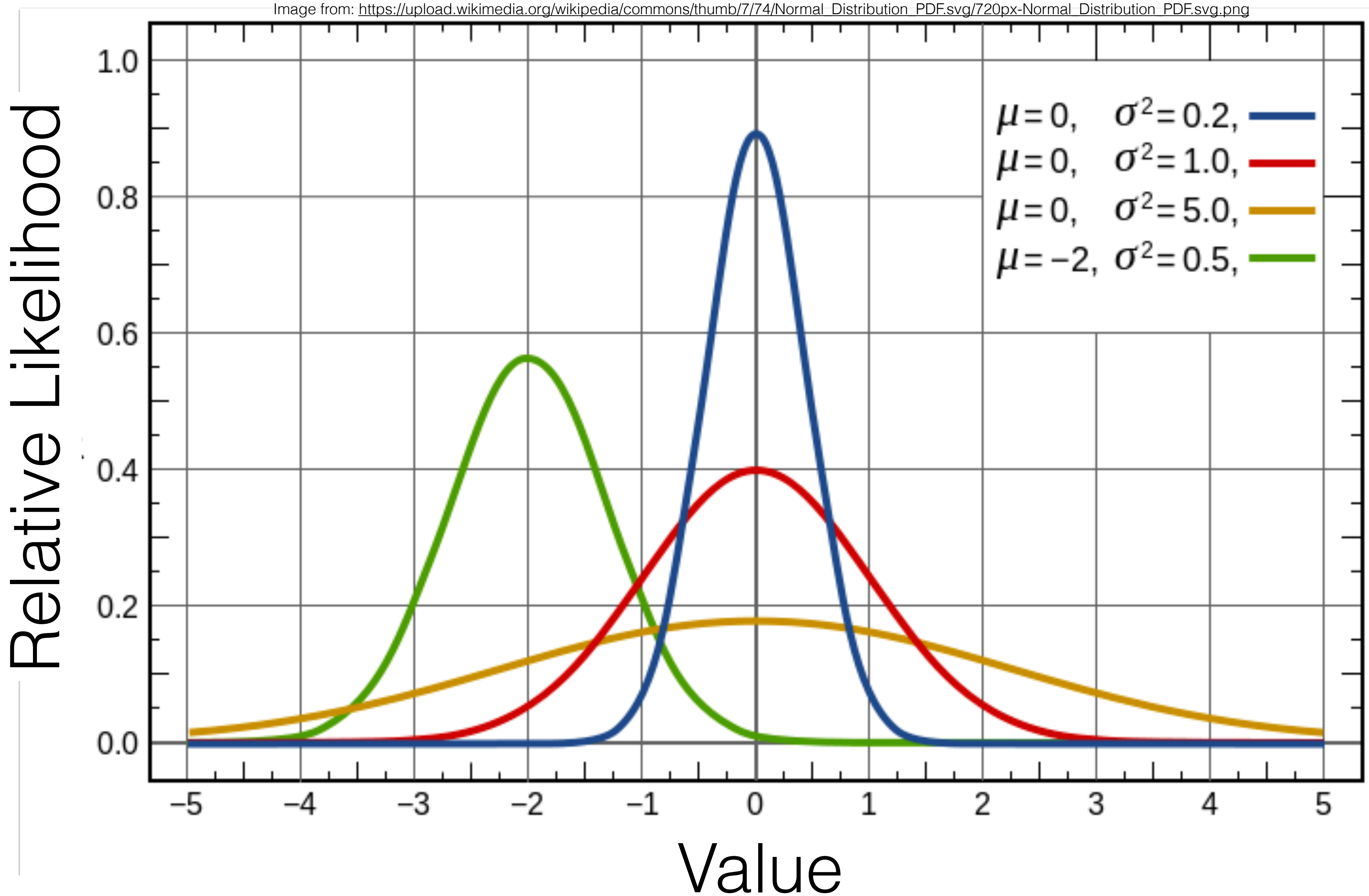
    equally likely

  - Pick the new value from a symmetric probability distribution centred at the current value.

Non-Uniform Mutation for Floating Point Representation

Normal Distribution $N(\mu,\sigma^2)$

Image from: https://upload.wikimedia.org/wikipedia/commons/thumb/7/74/Normal_Distribution_PDF.svg/720px-Normal_Distribution_PDF.svg.png



EA's Pseudocode

Evolutionary Algorithm

1. Initialise population
2. Evaluate each individual (determine their fitness)
3. Repeat (until a termination condition is satisfied)

    3.1 Select parents

    3.2 Recombine parents with probability Pc

    3.3 Mutate resulting offspring with probability Pm

    3.4 Evaluate offspring

    3.5 Select survivors for the next generation

EA's Pseudocode

Evolutionary Algorithm

1. Initialise population
2. Evaluate each individual (determine their fitness)
3. Repeat (until a termination condition is satisfied)

    3.1 Select parents

    3.2 Recombine parents with probability Pc

    3.3 Mutate resulting offspring with probability Pm

    3.4 Evaluate offspring

    3.5 Select survivors for the next generation

Survival Selection Mechanisms

- How many survivors?
  - Typically, if the population size is *S*, we will select *S* individuals to survive.
  - So, the population size is kept constant as generations pass.

- Types of survival selection:
  - Age-based selection.
  - Fitness-based selection.

Age-Based Survival Selection

- All offspring survive and all previous generation dies.
- Problem: may loose good individuals from the previous generation.

Fitness-Based Survival Selection

- Delete-worst:
  - Delete worse individuals among children + parents.
  - Only best individuals survive.
  - Problem: premature convergence.
- Elitism:
  - Frequently combined with age-based selection.
  - Always keep at least one copy of the fittest individual.
  - This copy replaces the worst child.

EA's Pseudocode

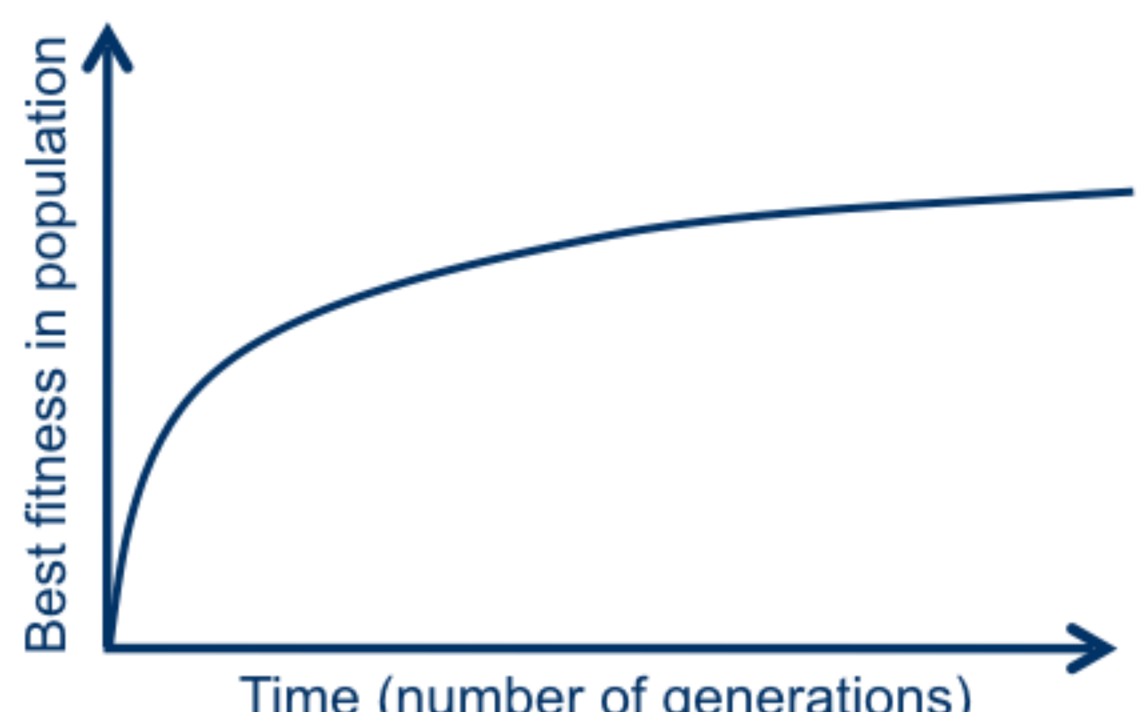
Evolutionary Algorithm

1. Initialise population
2. Evaluate each individual (determine their fitness)
3. Repeat **(until a termination condition is satisfied)**
    - 3.1 Select parents
    - 3.2 Recombine parents with probability Pc
    - 3.3 Mutate resulting offspring with probability Pm
    - 3.4 Evaluate offspring
    - 3.5 Select survivors for the next generation

Termination

- Reaching some maximum allowed number of generations.
- Reaching some (known/hoped for) fitness.
- Reaching some minimum level of diversity.
- Reaching some specified number of generations without fitness improvement.

Typical EA run:

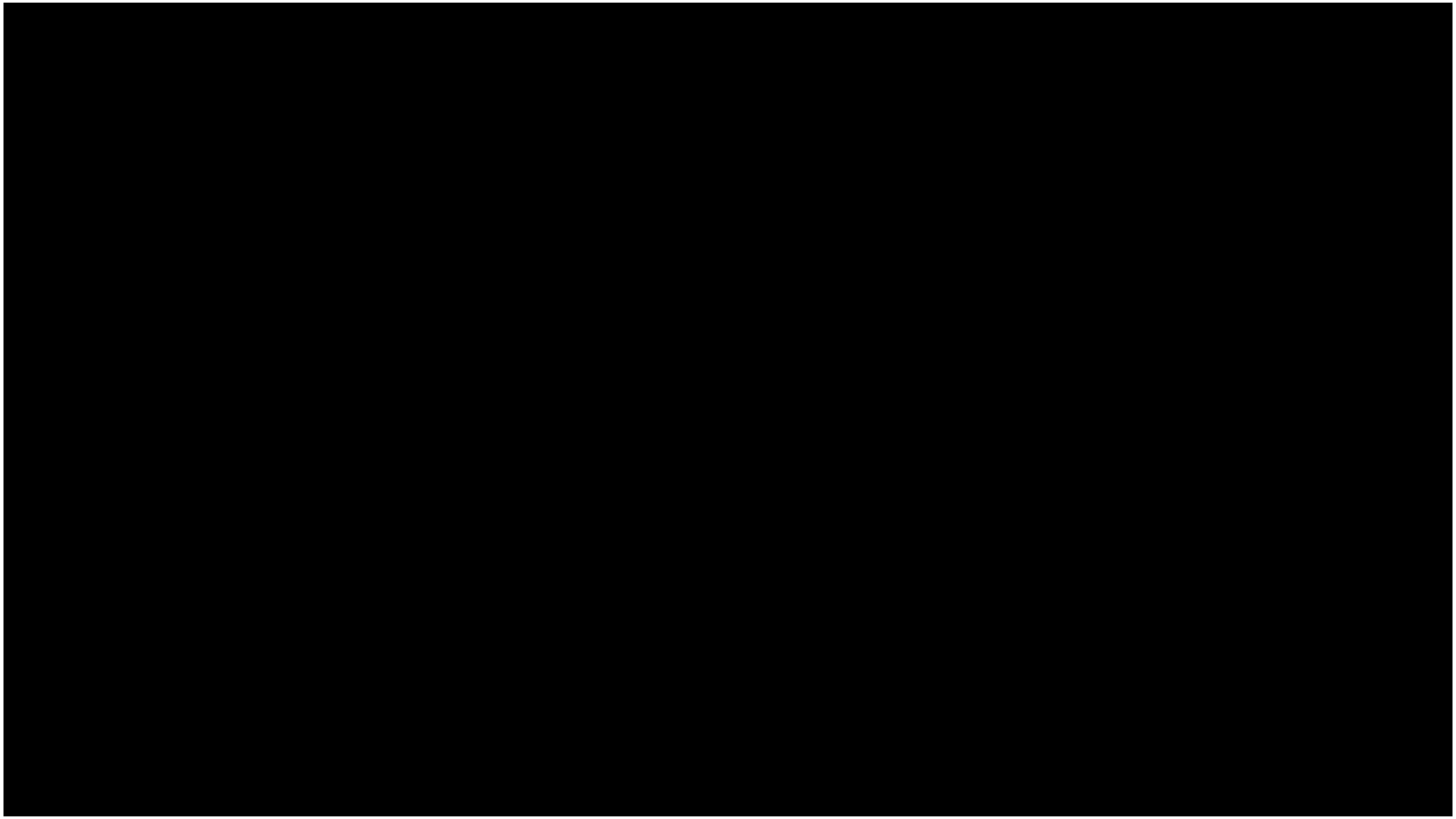


Designing an Evolutionary Algorithm

Representation
Initialisation
Recombination
Mutation
Parent selection
Survivor selection
Termination criteria
Fitness function
How to deal with constraints

Problem (In)Dependence

- Evolutionary algorithms can be applied to a variety of optimisation problems (problem independence).
- Choice of design depends on the problem.
  - Choice of representation depends on the problem.
  - Choice of recombination and mutation operators depends on the problem and the representation chosen.
  - …



[Youtube video (super mario bros) posted by Michael Roberts: https://youtu.be/05rEefXlmhI]

Summary of Variants of Evolutionary Algorithms

- Representation:
  - Binary strings
  - Integer vectors
  - Floating-point vectors
  - Permutations
  - Matrices
  - Etc
- Parents selection:
  - Roulette wheel
  - Tournament selection
  - Ranking selection
- Crossover for binary and integer representation
  - 1-Point Crossover
  - N-Points Crossover
  - Uniform
- Crossover for floating point representation
  - Discrete
  - Intermediate
  - There are also recombination methods for more than 2 parents
- Crossover for permutations
  - Order 1 crossover
  - Partially mapped crossover
  - Cycle crossover
  - Edge recombination
- Mutation for binary representation
  - Bitwise bit-flipping
- Mutation for integer representation
  - Random reset
  - Creep mutation
- Mutation for floating-point representation
  - Uniform
  - Non-uniform
- Mutation for permutations
  - Swap mutation
  - Insert Mutation
  - Inversion Mutation
  - Scramble Mutation
- Survival selection:
  - Age-based
    - Generational
  - Fitness-based
    - Delete-worst
    - Elitism

Evolutionary Algorithm Glossary

- Population = multiset of individuals.
- Individual = candidate solution
- Chromosome = representation of candidate solution
- Gene = a component of chromosome
- Allele = value at a gene
- Mutation = unary variation operator
- Crossover = binary variation operator
- Recombination = n-ary variation operator
- Fitness function = objective or quality function
- Generations = iterations

Further Reading

- Eiben and Smith, Introduction to Evolutionary Computing, Chapter 3 (Genetic Algorithms), Springer 2003.