COMP 3200
Artificial Intelligence

Lecture 14

Intro to Evolutionary Computing

Evolutionary Algorithms

Positioning of EC

- EC is part of computer science
- EC is not part of life sciences / biology
- Biology delivers inspiration + terminology
- EC can be applied in biological research, but has many possible applications

The Main EC Metaphor

Problem Solving	Evolution
Problem	Environment
Candidate Solution	Individual
Quality	Fitness

- Quality = chance for seeding new solution
- Fitness = chance of survival / reproduction

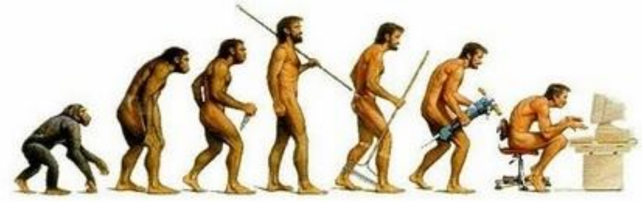
Brief History of EC

- 1948, Turing: “genetical or evolutionary search”
- 1962, Bremermann: optimization through evolution
- 1964, Rechenberg: evolutionary strategies
- 1965, Fogel: evolutionary programming
- 1975, Hollan: genetic algorithms
- 1992, Koza: genetic programming

- 3200: Assignment 4!

Darwinian Evolution: Survival of the Fittest

- All environments have finite resources
- Life forms have basic instinct / life cycles geared toward reproduction
- Therefore, some sort of selection inevitable
- Individuals that compete for resources most effectively have increased chance of reproduction
- Note: ‘Fitness’ in nature is a derived, secondary measure. ie: we assign a high fitness to individuals with many offspring



Darwinian Evolution

- Phenotypic Traits
  - Behaviours / physical differences that affect individual responses to the environment
  - Partly determined by inheritance, partly by factors during development (nature/nurture)
  - Unique to each individual, partly as a result of random changes
- Trait Inheritance
  - If phenotypic traits lead to higher chances of reproduction, then these traits are passed on to offspring (inherited)
  - Along with random mutations, this leads to new combinations of traits that lead to more 'fit' individuals (ie: more offspring)

Darwinian Evolution

- Population consists of many (possibly diverse) individuals

- Combinations of traits that are better suited for a given environment lead to higher chance of reproduction
  - Individuals are "unit of selection"

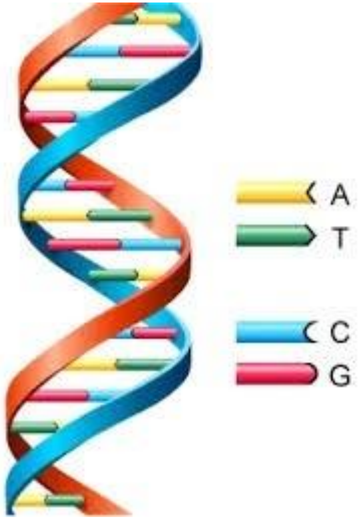
- Variations occurring through random changes yield constant source of diversity, coupled with selection means:
  - Population is the "unit of evolution"

Genetics

- WARNING: *I AM NOT A BIOLOGIST*
- The information required to build a living organism is coded in the organism's DNA
- Genotype (DNA inside) determines phenotype
- Genotype to phenotypic traits is a complex mapping
  - One gene may affect many traits (pleiotropy)
  - Many genes may affect one trait (polygeny)
- Changes in the genotype may lead to changes in the organism (height, hair color)

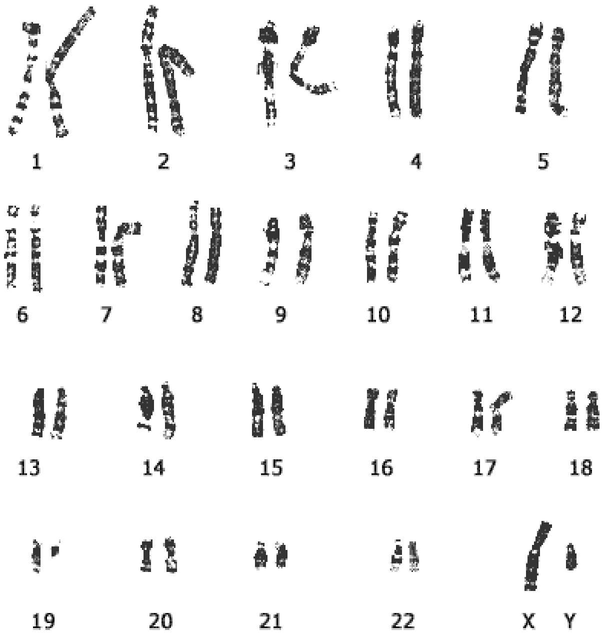
Genes and the Genome

- Deoxyribonucleic Acid (DNA) and nitrogenous bases
  - Adenine, Thymine, Cytosine, Guanine
- Genes are functional unit of stretches of DNA on chromosomes
- The complete genetic material in an individual’s genotype is called the genome



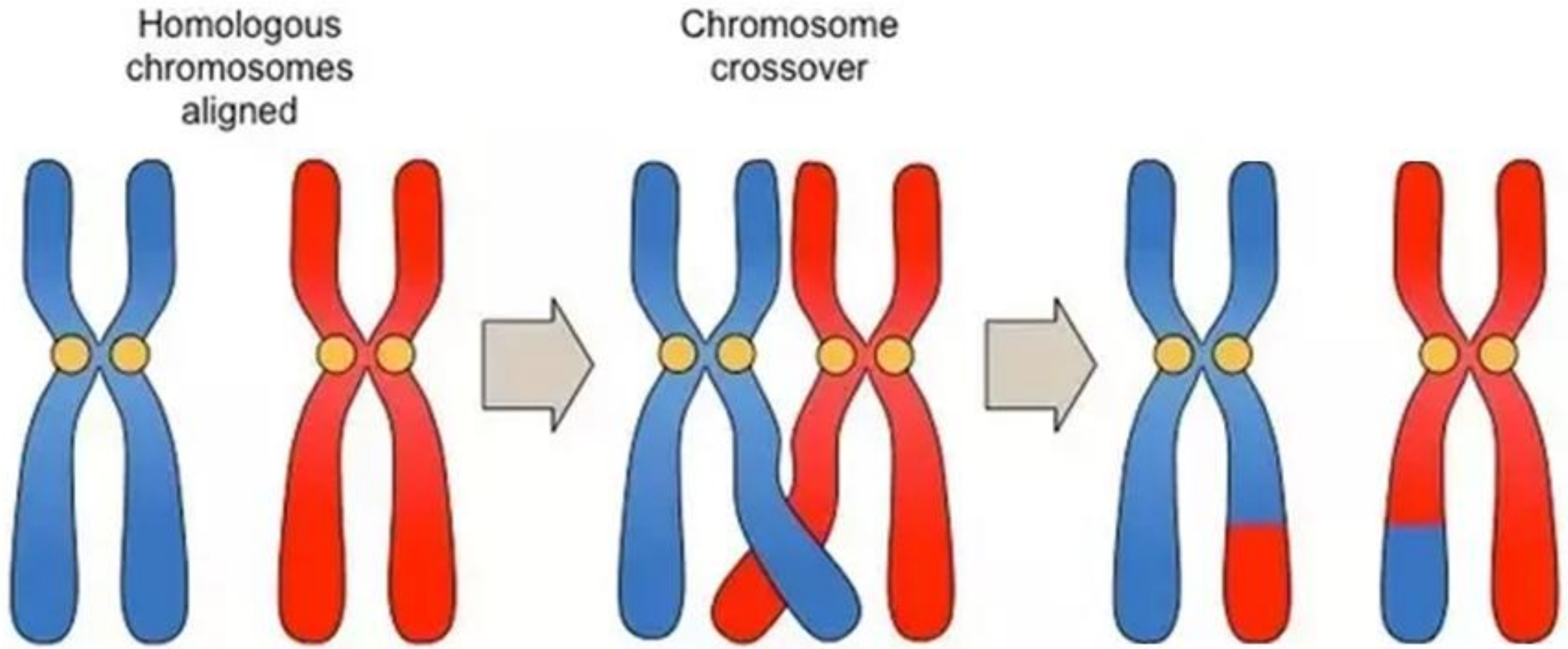
Example: Homo Sapiens

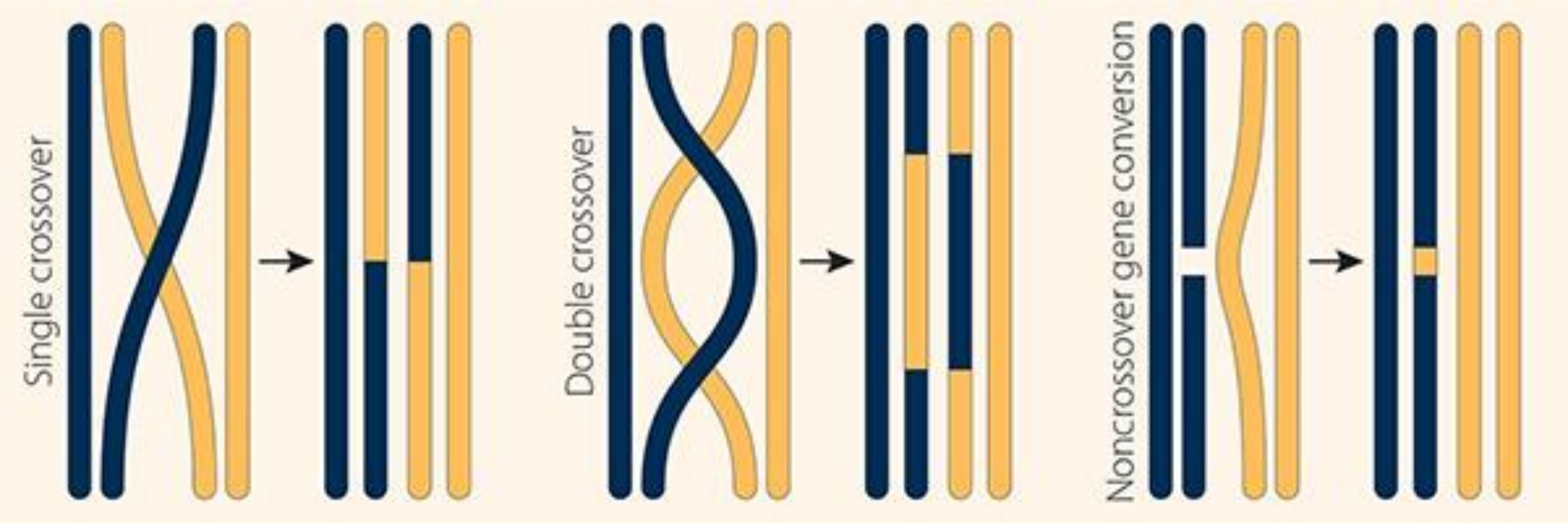
- Human DNA is organized into chromosomes
- Human body cells contain 23 pairs of chromosomes which together define the physical attributes of the individual

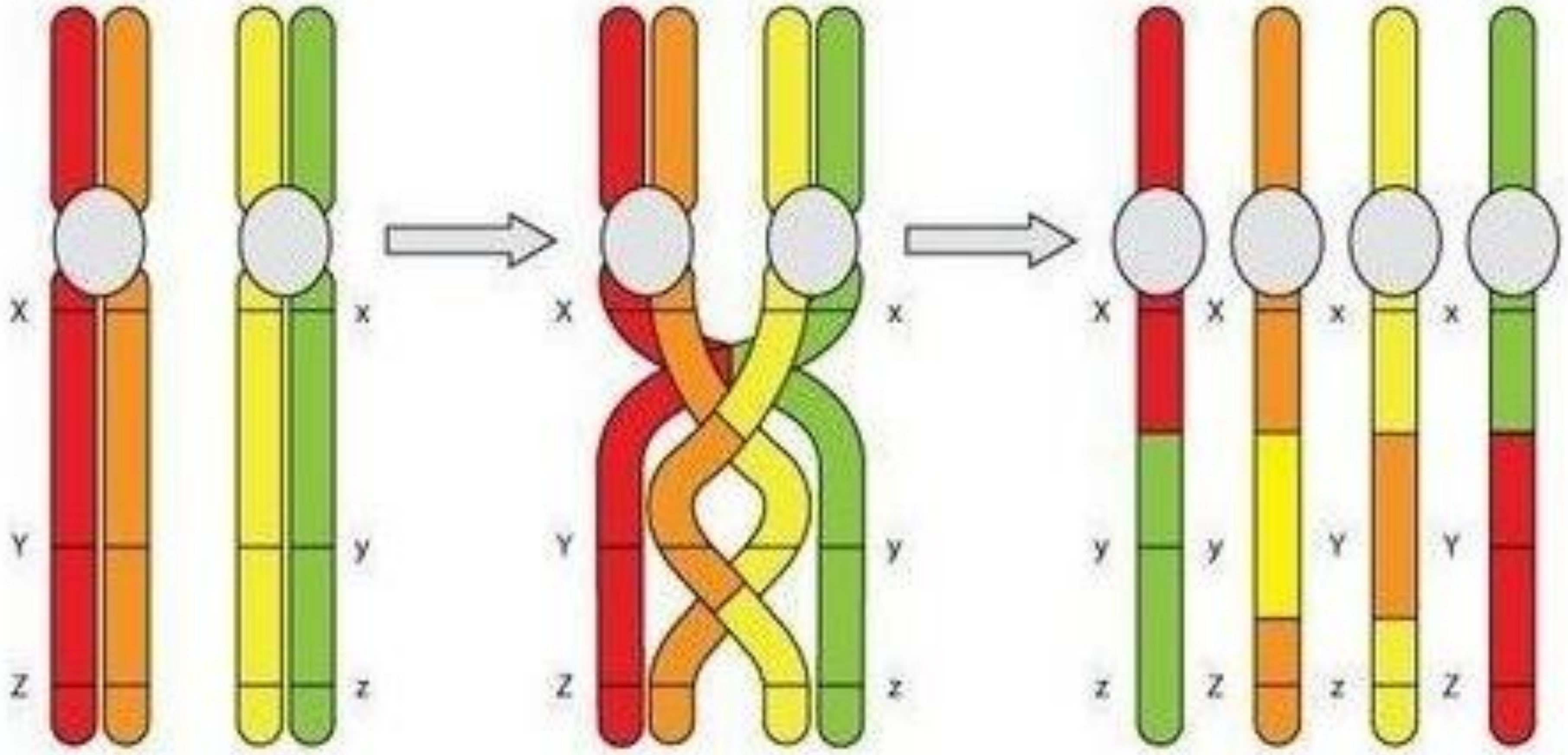


Reproductive Cells

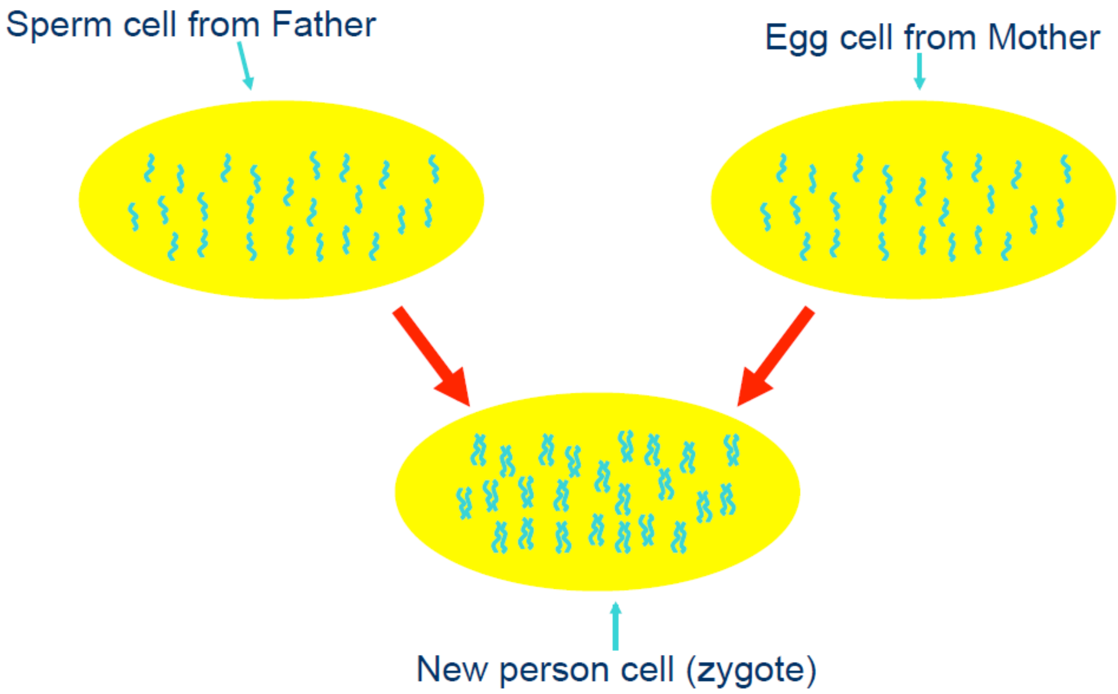
- Gametes (sperm and egg cells) contain 23 individual chromosomes rather than 23 pairs
- Gametes are formed by a special form of cell splitting called meiosis
- During meiosis, the pairs of chromosome undergo an operation called cross-over
- Crossover shares genetic information from both parents to form offspring


Homologous
chromosomes
aligned
Chromosome
crossover


Single crossover
Double crossover
Noncrossover gene conversion

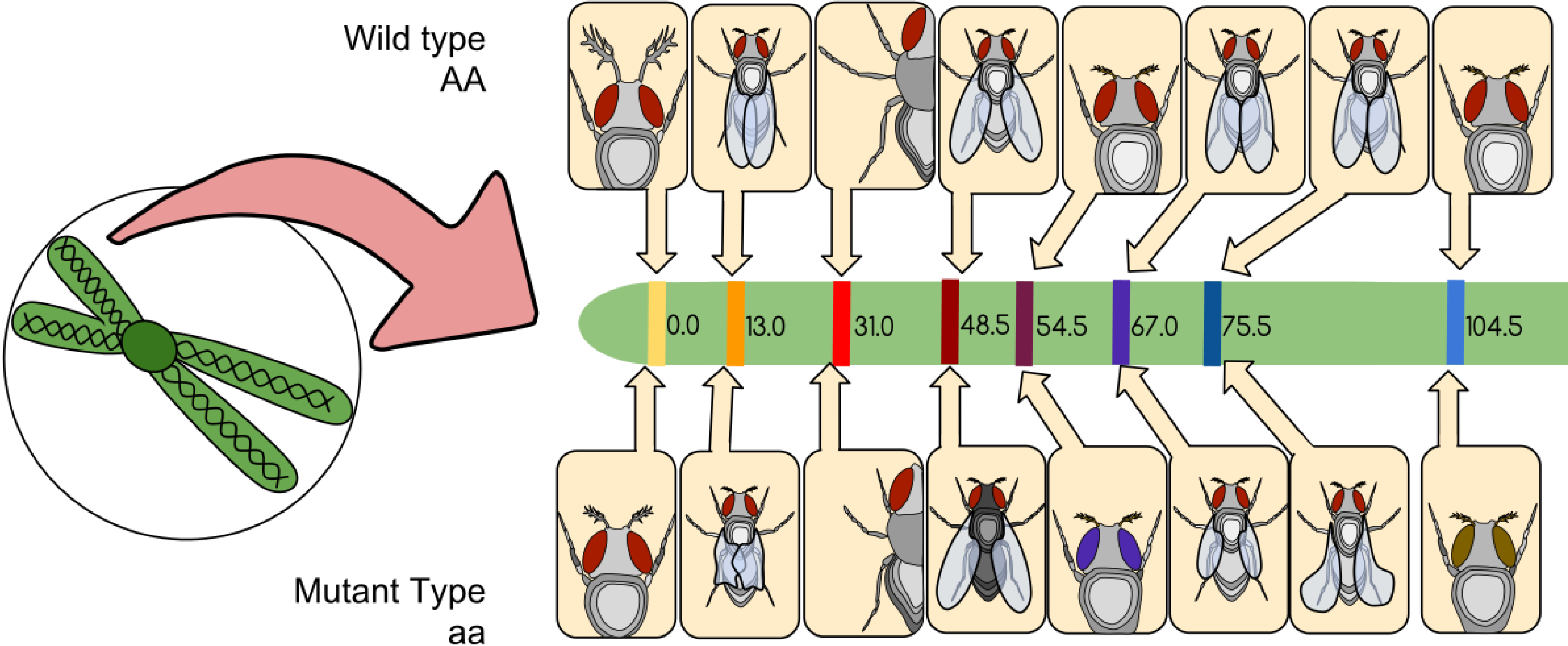

X
Y
Z
x
y
z
X
Y
Z
x
y
z
X
y
z
X
y
Z
x
Y
z
x
Y
Z

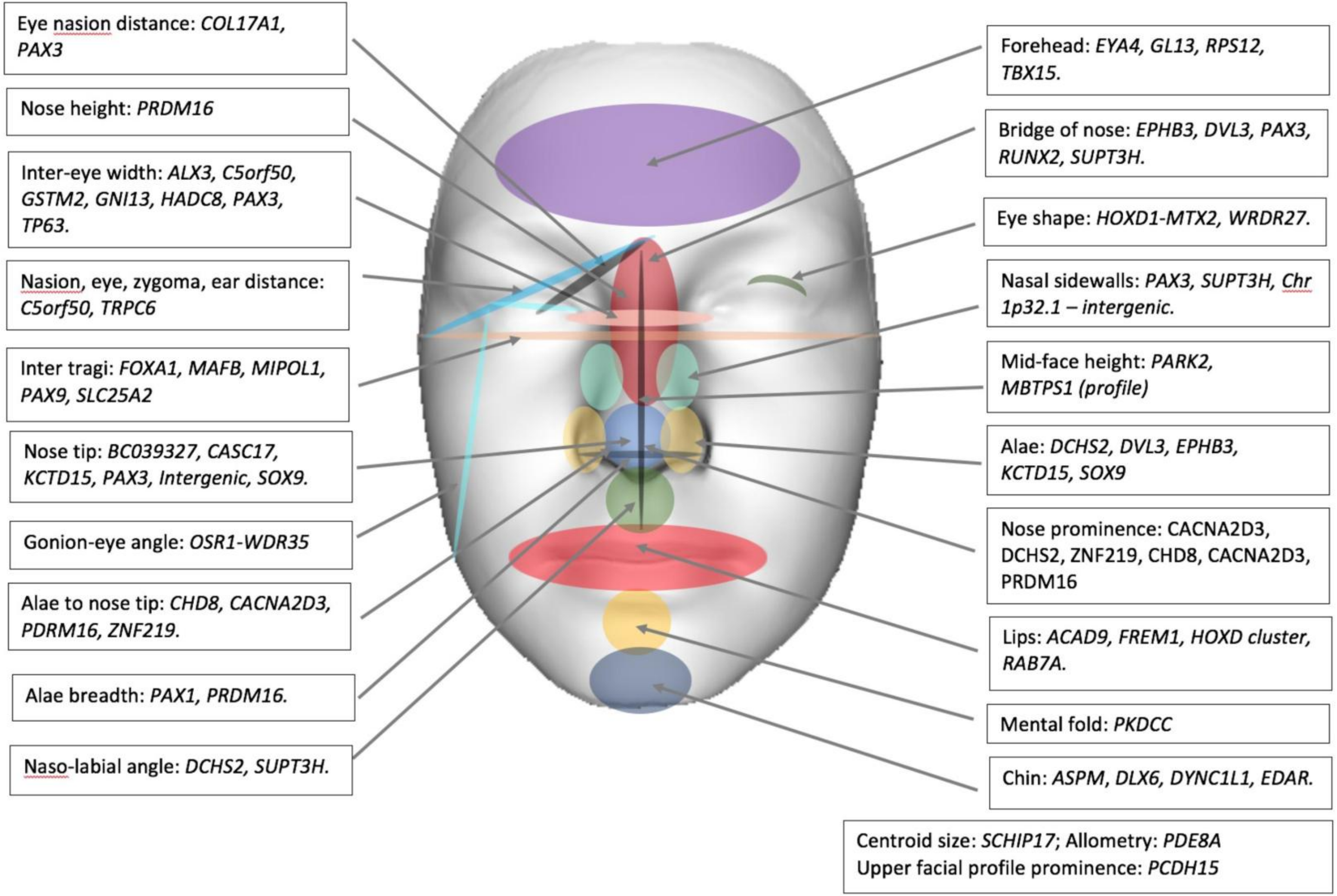
Fertilization



Genetic Code

- All proteins in life on earth are composed of sequences built from 20 different amino acids
- DNA is built from four nucleotides in a double helix spiral: Purines A, G and Pyrimidines T, C
- Triplets of these form codons, each of which codes for a specific amino acid
- Genetic code = the mapping from codons to amino acids


Wild type
AA
Mutant Type
aa
0.0
13.0
31.0
48.5
54.5
67.0
75.5
104.5


Eye nasion distance: *COL17A1, PAX3*
Nose height: *PRDM16*
Inter-eye width: *ALX3, C5orf50, GSTM2, GNI13, HADC8, PAX3, TP63.*
Nasion, eye, zygoma, ear distance: *C5orf50, TRPC6*
Inter tragi: *FOXA1, MAFB, MIPOL1, PAX9, SLC25A2*
Nose tip: *BC039327, CASC17, KCTD15, PAX3, Intergenic, SOX9.*
Gonion-eye angle: *OSR1-WDR35*
Alae to nose tip: *CHD8, CACNA2D3, PDRM16, ZNF219.*
Alae breadth: *PAX1, PRDM16.*
Naso-labial angle: *DCHS2, SUPT3H.*
Forehead: *EYA4, GL13, RPS12, TBX15.*
Bridge of nose: *EPHB3, DVL3, PAX3, RUNX2, SUPT3H.*
Eye shape: *HOXD1-MTX2, WRDR27.*
Nasal sidewalls: *PAX3, SUPT3H, Chr 1p32.1 – intergenic.*
Mid-face height: *PARK2, MBTPS1 (profile)*
Alae: *DCHS2, DVL3, EPHB3, KCTD15, SOX9*
Nose prominence: CACNA2D3, DCHS2, ZNF219, CHD8, CACNA2D3, PRDM16
Lips: *ACAD9, FREM1, HOXD cluster, RAB7A.*
Mental fold: *PKDCC*
Chin: *ASPM, DLX6, DYNC1L1, EDAR.*
Centroid size: *SCHIP17*; Allometry: *PDE8A*
Upper facial profile prominence: *PCDH15*

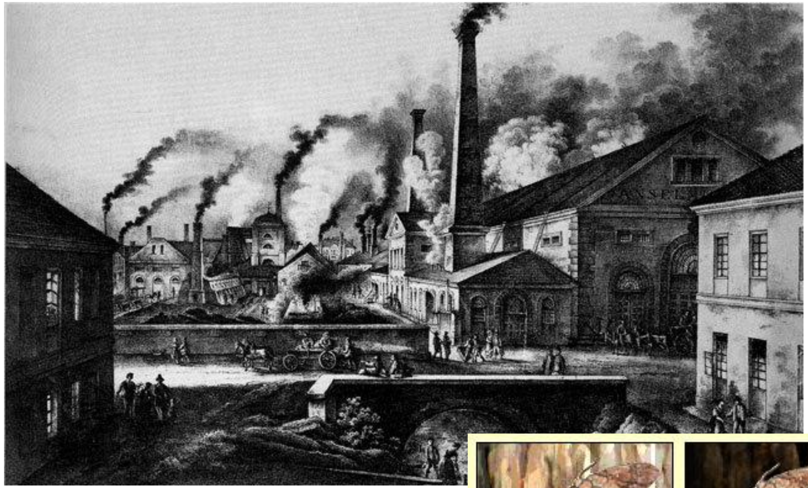
Mutation

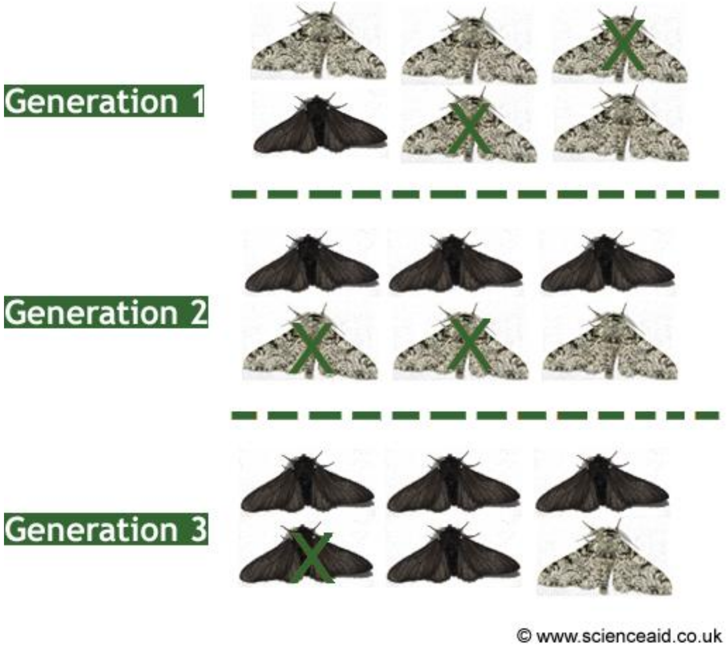
- Occasionally, some of the genetic material changes very slightly during this process
- Caused by replication error, environment
- This means the child might have genetic material not inherited from either parent
- This can be
  - Catastrophic: offspring not viable (probable)
  - Neutral: new feature doesn't influence fitness
  - Advantageous: strong new feature occurs

Important Evolution Notes

- Individuals do not *intentionally change* themselves to suit an environment, there is no *learning* involved in the process
- Fit individuals reproduce, unfit ones don't
- Good traits of parents passed to offspring, producing individuals fitter than either parent
- Random mutation can introduce new traits
- This process produces more fit populations

Evolution: Peppered Moth





Motivation for EC

- Nature has always served as inspiration for engineers and scientists
- Developing new problem solving methods (algorithms) is a central theme in math and CS
- Complexity of problems to be solved increases
- Robust problem solving technology is required

Motivation for EC

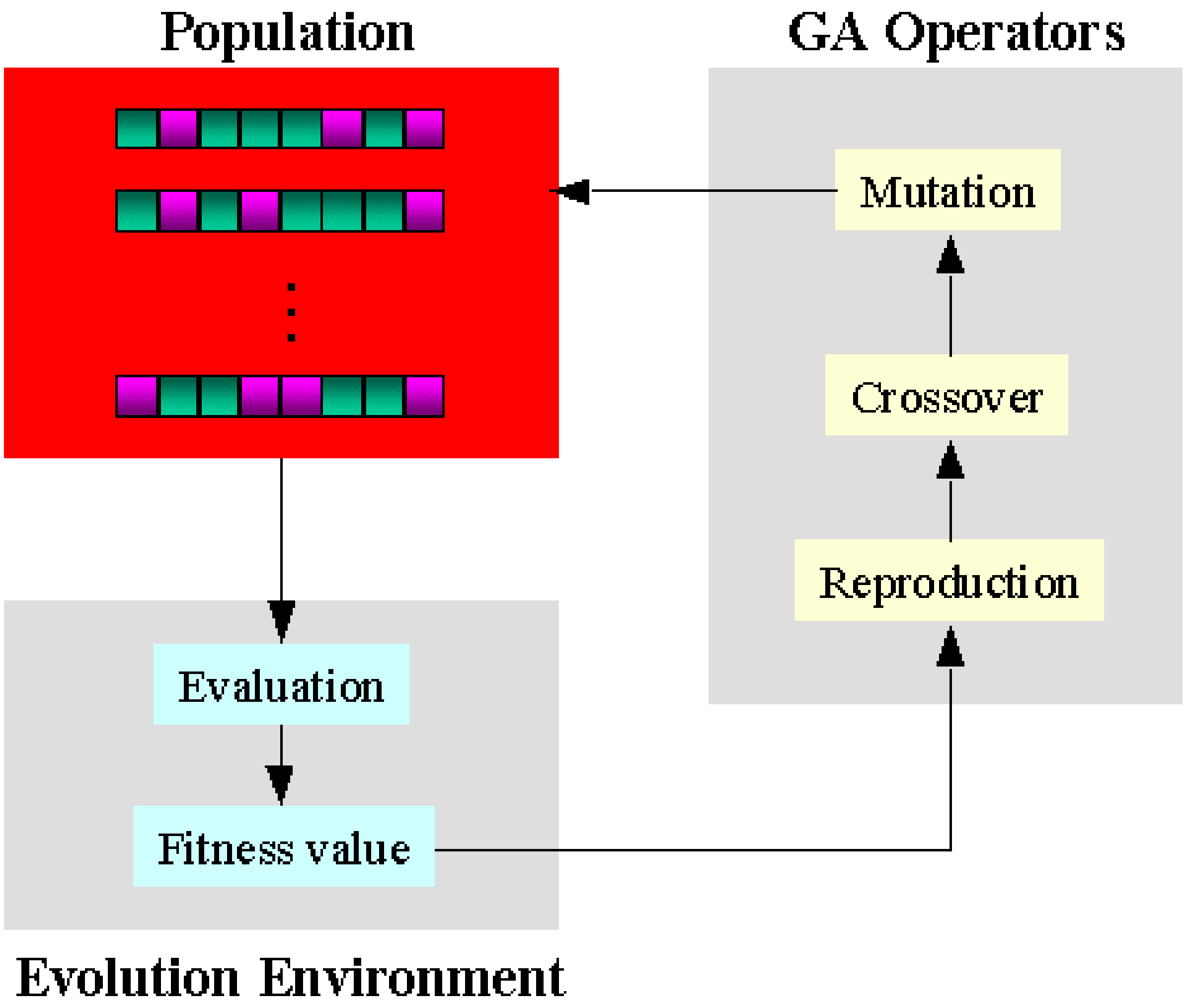
- Problems too complex for existing algorithms
- Use evolution as problem solving algorithm
- Evolutionary Computing can simulate evolutionary process with millions of generations
- If we can model the problem in terms of environment, individual, fitness, perhaps evolutionary computation can provide solutions

Example Problem: Exam Scheduling

- Problem Components
  - Profs, Students, Rooms, Courses, Time Slots
- Constraints to Satisfy
  - Student/Prof have <= 1 exam at a time
  - No room has > 1 exam in it at a time
  - Student has < 3 exams in a day
- Gigantic search space, majority of not valid
- EC: Schedule = Individual, Fitness = Validity

EC Metaphor

1. Population of individuals exists in an environment with limited resources
2. Competition for those resources causes selection of fitter individuals that are better adapted
3. Those individuals reproduce to form new generation of individuals through recombination and mutation
4. New individuals have fitness evaluated, high fitness individuals chosen to reproduce, pass on good traits
5. Over time, natural selection causes fitness to rise


Population
GA Operators
Mutation
Crossover
Reproduction
Evaluation
Fitness value
Evolution Environment

Evolutionary Algorithms

1. INITIALIZE population w/ random individuals
2. REPEAT UNTIL (termination condition)
3. EVALUTE population / individual fitness
4. SELECT parents with high fitness
5. COMBINE parents to form offspring
6. MUTATE resulting offspring
7. NEXT POPULATION = offspring

Different Types of EA

- Different EA have different representations
  - Binary strings (Integers): Genetic Algorithms
  - Real-valued vectors: Evolution Strategies
  - Finite state machines: Evolutionary Programming
  - Tree Structure: Genetic Programming
- Differences largely cosmetic, best to
  - Choose representation to suit problem
  - Choose variation operators to suit representation

Main Components of an EA

- Representation (definition of individuals)
- Evaluation / Fitness Function
- Population (Size, Shape)
- Parent Selection Mechanism
- Variation Operators (Recombination / Mutation)
- Survivor Selection Mechanism (Replacement)

Representations

- Candidate solutions (individuals) exist in phenotype space
    - Phenotype = Actual Solution Candidate
- They are encoded in chromosomes, exist in genotype space
    - Genotype = Representation of Phenotype
    - Encoding: Phenotype -> Genotype
    - Decoding: Genotype -> Phenotype (1 to 1)
- In order to find global optimum, every possible solution must be represented in genotype space

Representation Example

Phenotype

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

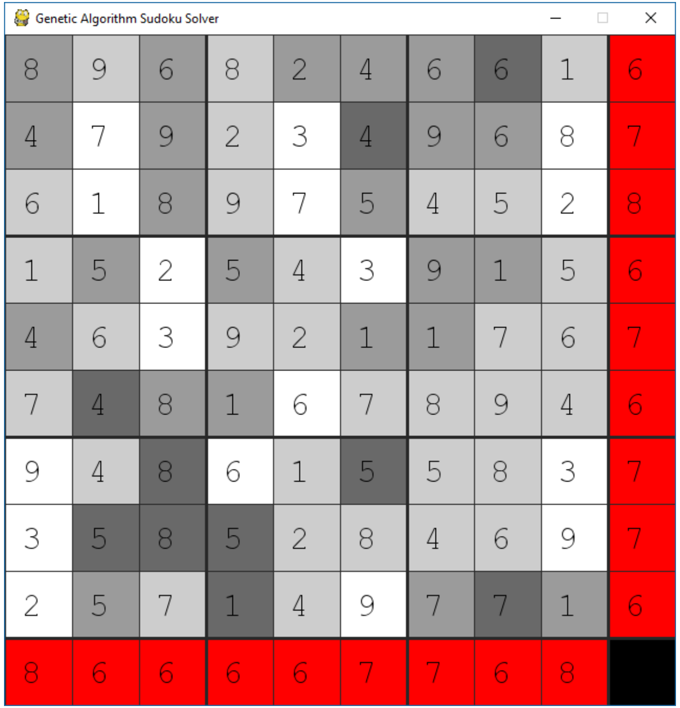
Genotype

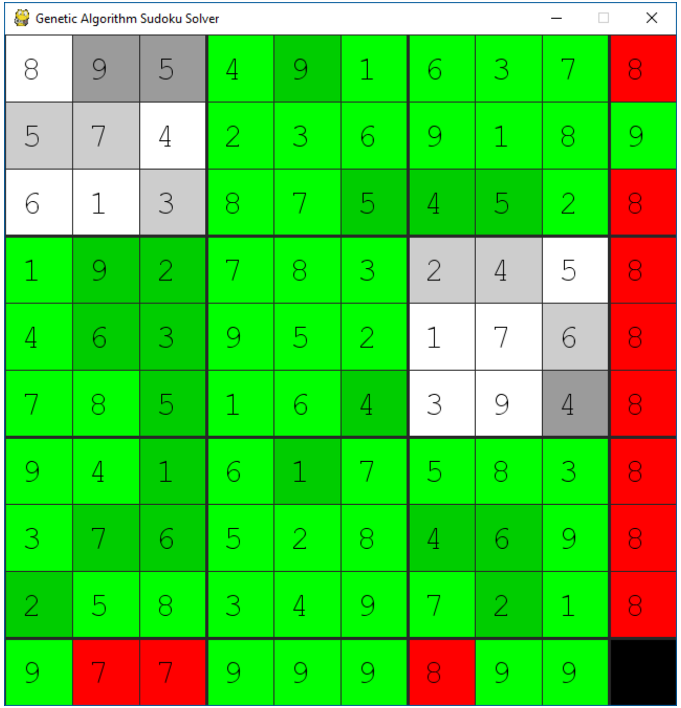
[5,3,0,0,7,0,0,0,0,…0,7,9]

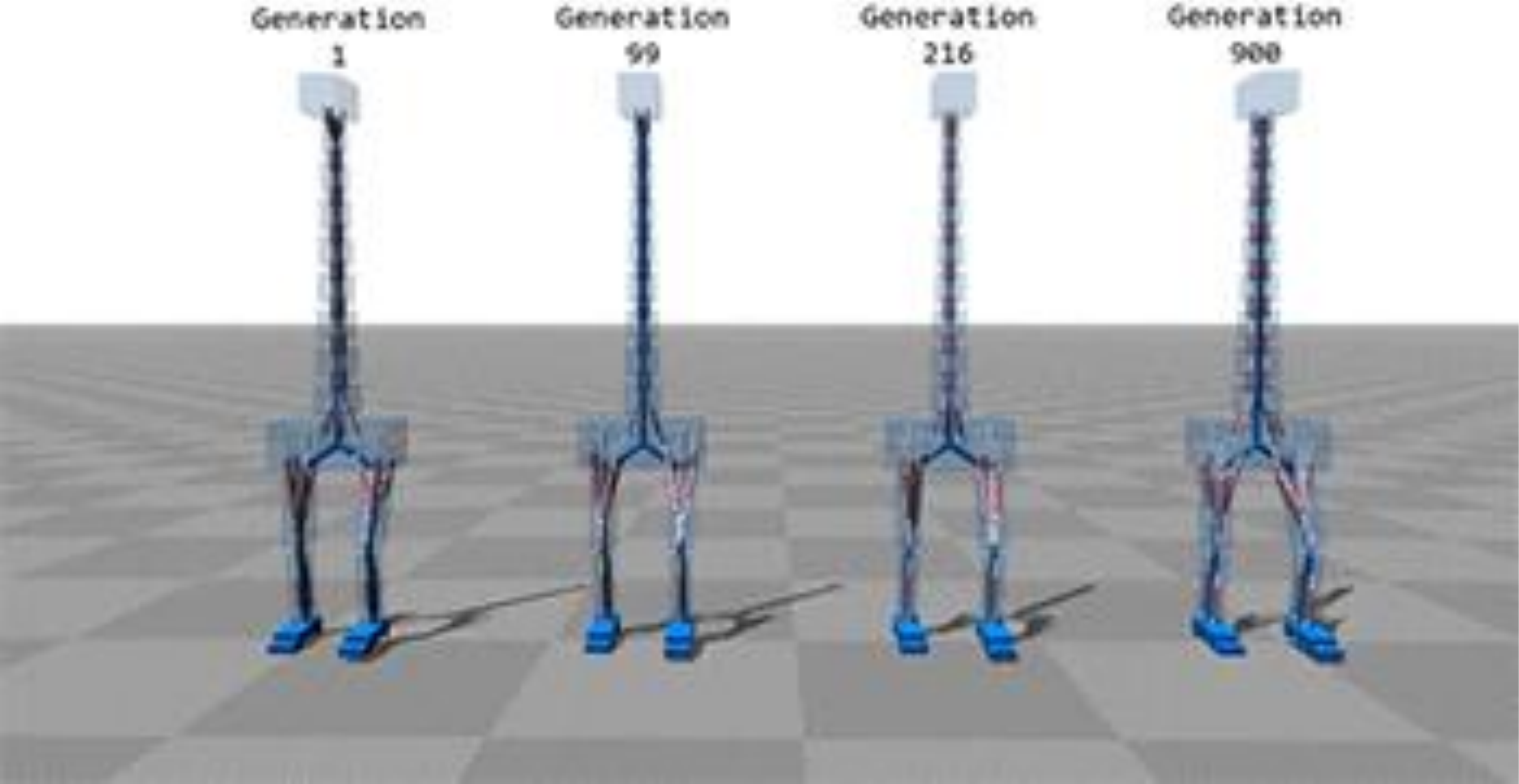
Evaluation (Fitness) Function

- Represents an estimate of how well an individual will perform in a given environment
  - How 'fit' they are to reproduce
- Assigns a real-valued fitnesses to each phenotype which forms the basis for selection
  - More fine-grained different values, the better
- Usually we talk about fitness being maximized
  - Some problems may be minimization

Example Fitness Function

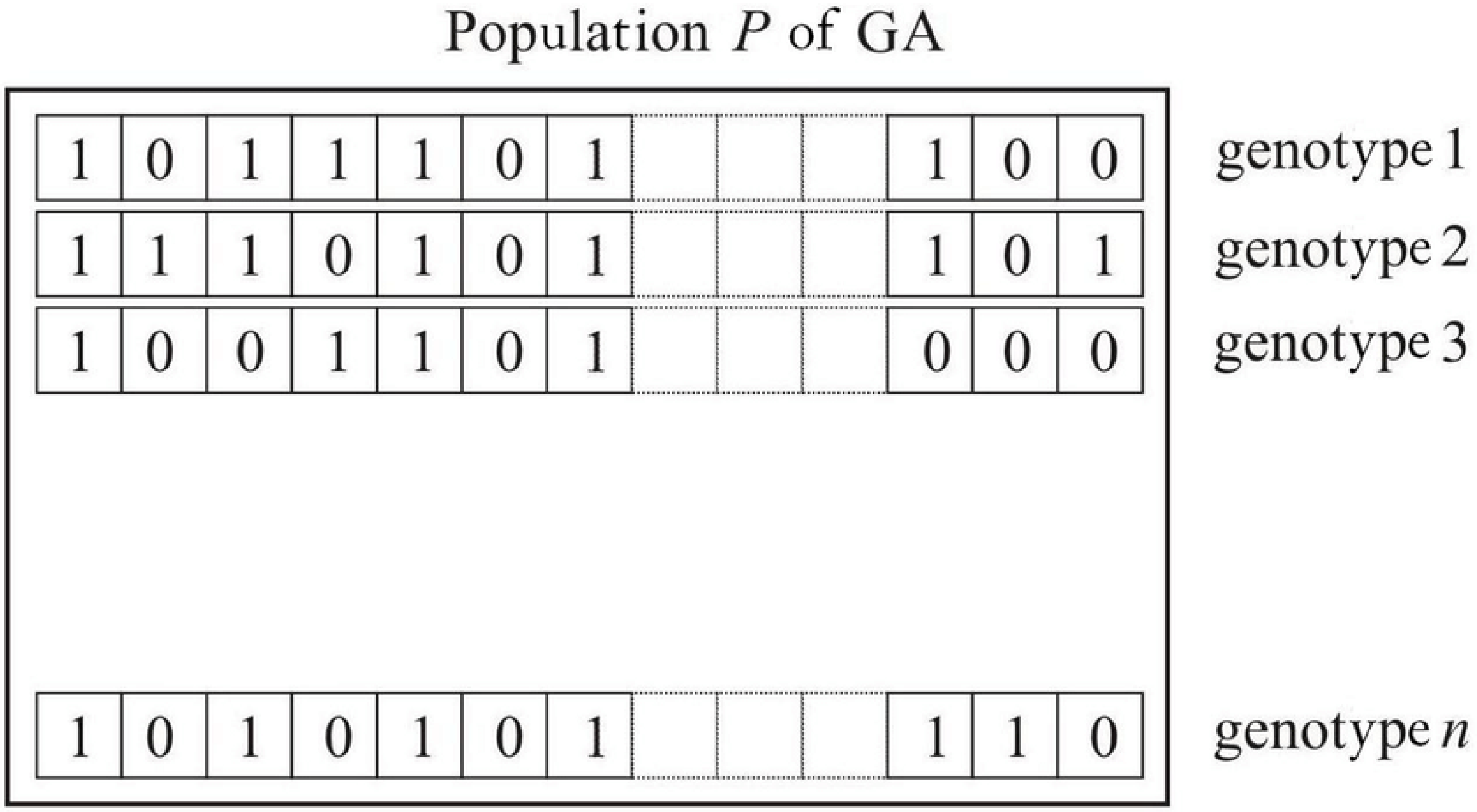





Generation
1
Generation
99
Generation
216
Generation
900

Population

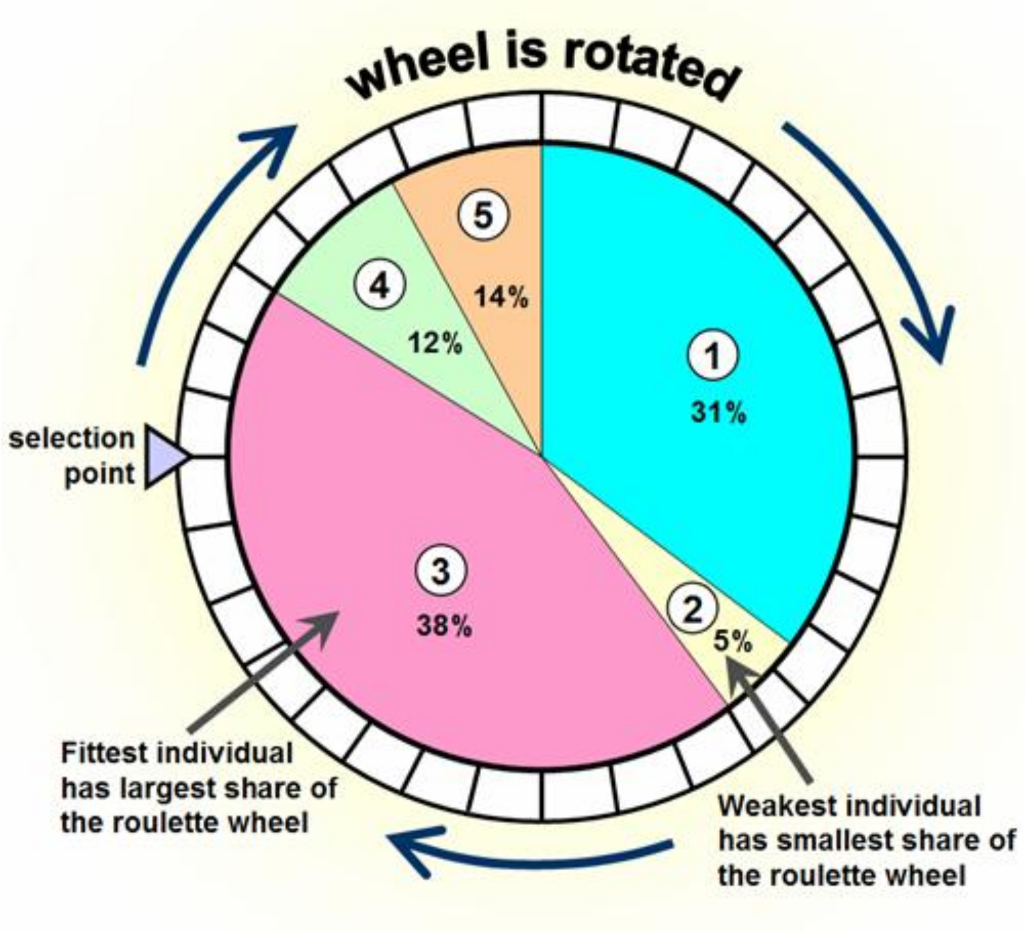
- Holds (representation of) possible solutions
- Usually has a fixed size
- Some sophisticated EAs assert a special structure on the population (grid, etc)
- Selection operators usually take whole population into account (current generation)
- Diversity of population refers to the number of different fitnesses / phenotypes / genotypes


Population P of GA
1 0 1 1 1 0 1 1 0 0
genotype 1
1 1 1 0 1 0 1 1 0 1
genotype 2
1 0 0 1 1 0 1 0 0 0
genotype 3
1 0 1 0 1 0 1 1 1 0
genotype n

Parent Selection Mechanism

- Assigns variable probabilities of individuals as parents depending on their fitnesses
- Usually Probabilistic
  - High quality solutions more likely to reproduce (be a parent) but not guaranteed
  - Worst candidate usually has non-zero chance
- Stochastic nature can help escape local optima

Example Parent Selection

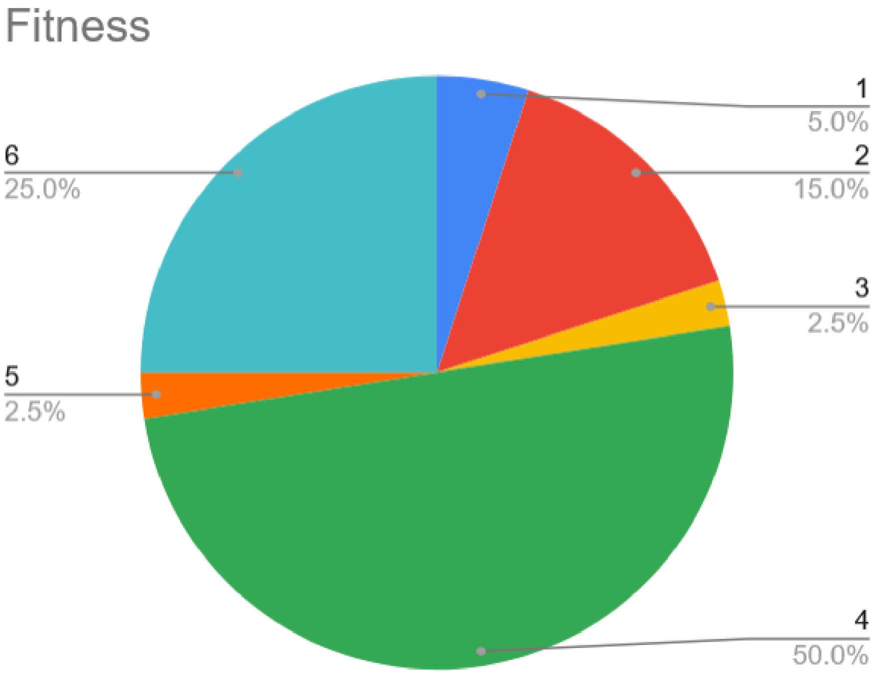


Example Parent Selection

Example Parent Selection

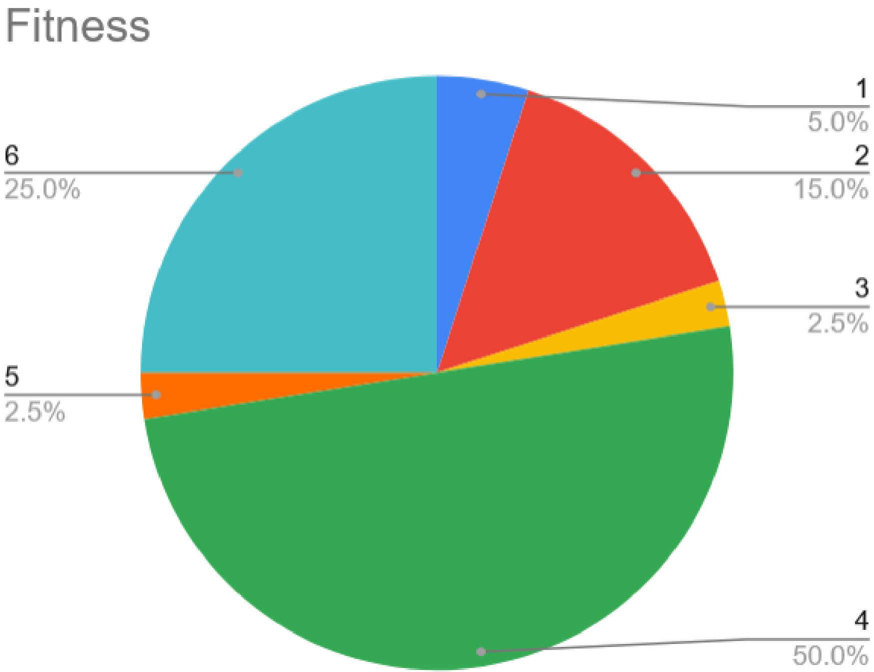
Roulette Wheel Selection

Population	Fitness	Sum
Individual 1	10	10
Individual 2	30	40
Individual 3	5	45
Individual 4	100	145
Individual 5	5	150
Individual 6	50	200



Roulette Wheel Selection

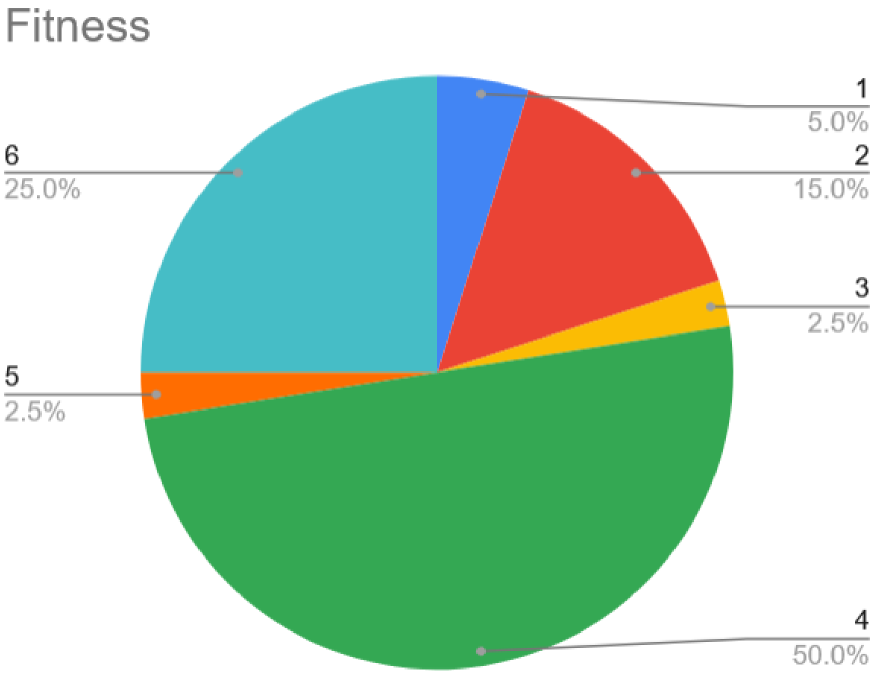
Population	Fitness	Sum
Individual 1	10	10
Individual 2	30	40
Individual 3	5	45
Individual 4	100	145
Individual 5	5	150
Individual 6	50	200



Pick Random Number
R from 1 to sum[end]

Roulette Wheel Selection

Population	Fitness	Sum
Individual 1	10	10
Individual 2	30	40
Individual 3	5	45
Individual 4	100	145
Individual 5	5	150
Individual 6	50	200



Check sum values until sum[i] >= R

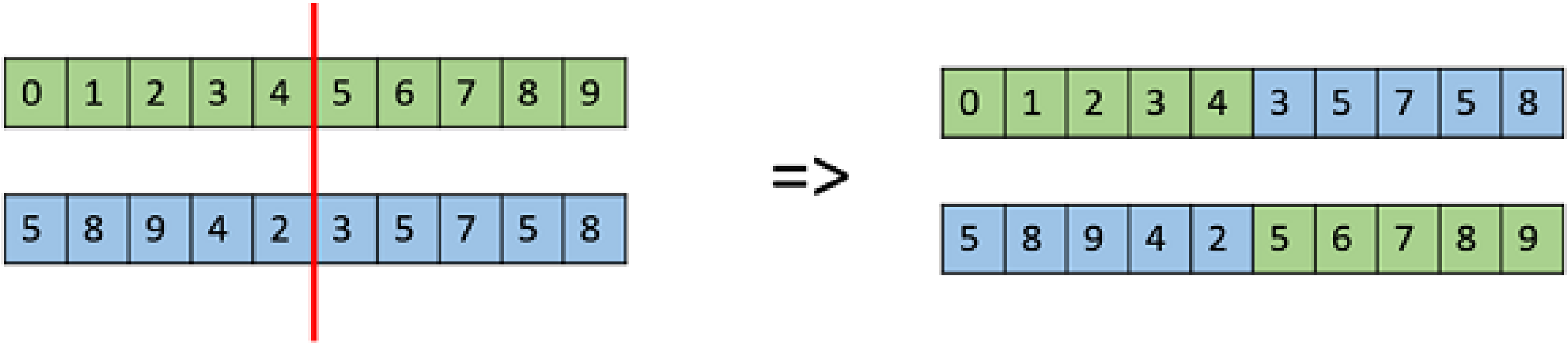
Roulette Wheel Selection

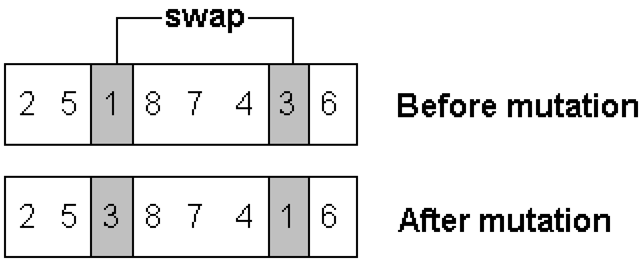
```
roulette_select(population)
    pop_fitness = [fit(p1), fit(p2), …, fit(pn)]
    sum = sum(pop_fitness)
    pick = random(0, sum)
    current = 0
    for (i=0; i<population.length; i++):
        current += pop_fitness[i]
        if (current > pick): return population[i]
```

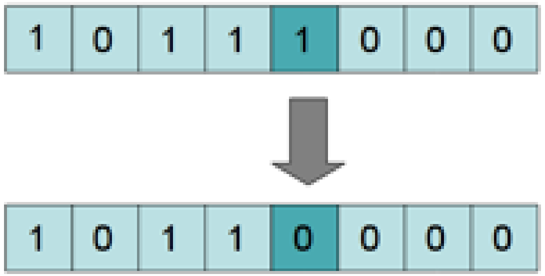
Variation Operators

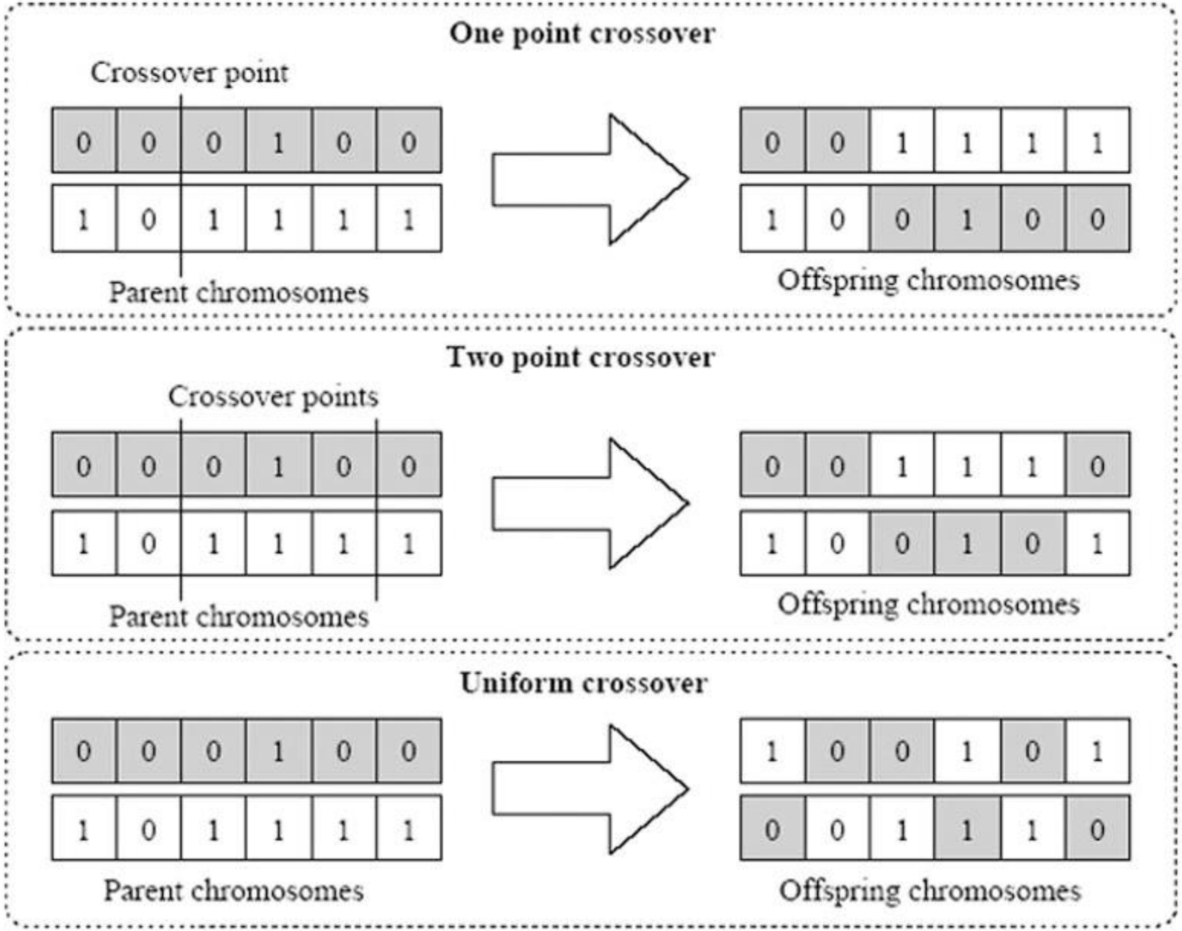
- Role is to generate new candidate solutions
  - From parents to children (offspring)
- Usually divided into two types according to their number of inputs:
  - Mutation Operator (1 input)
  - Recombination Operator (> 1 input)
  - 2 Inputs = Crossover
- Most EA use both recombination and mutation

Crossover / Mutation







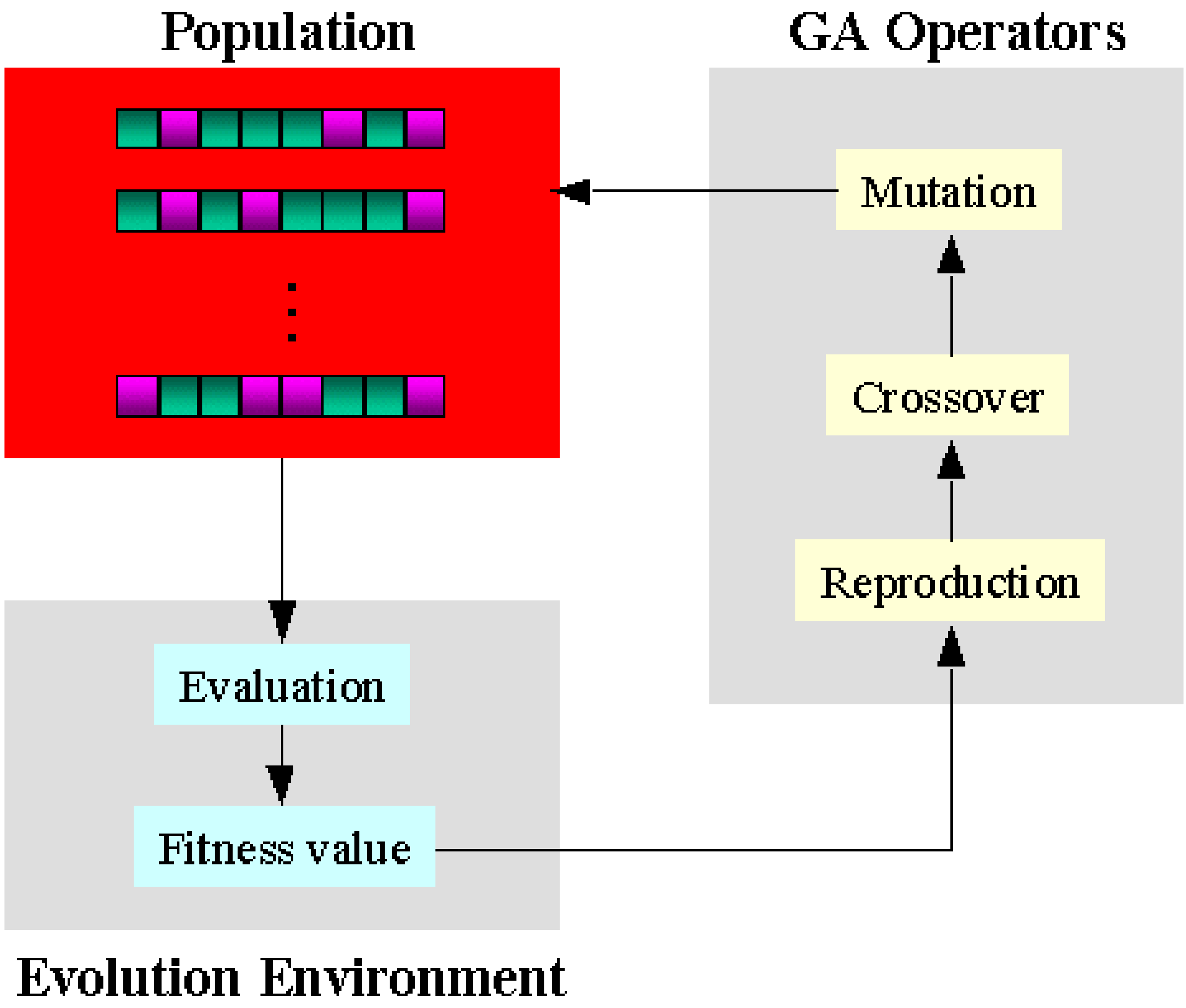

One point crossover
Crossover point
0 0 0 1 0 0
1 0 1 1 1 1
Parent chromosomes
0 0 1 1 1 1
1 0 0 1 0 0
Offspring chromosomes
Two point crossover
Crossover points
0 0 0 1 0 0
1 0 1 1 1 1
Parent chromosomes
0 0 1 1 1 0
1 0 0 1 0 1
Offspring chromosomes
Uniform crossover
0 0 0 1 0 0
1 0 1 1 1 1
Parent chromosomes
1 0 0 1 0 1
0 0 1 1 1 0
Offspring chromosomes

Survivor Selection

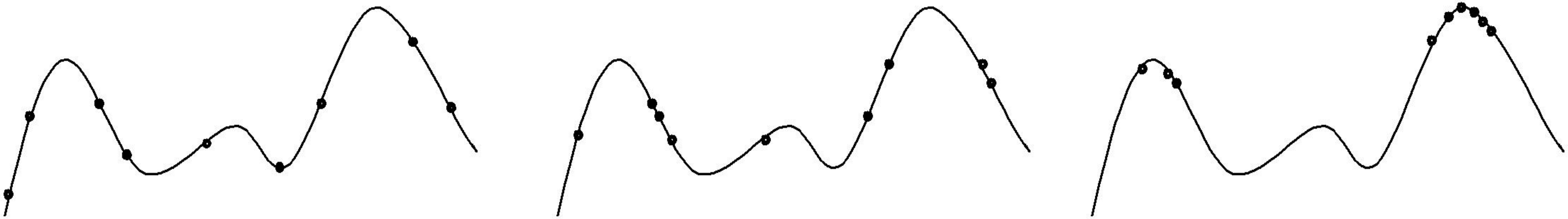
- Also called environmental selection
- Most EA use a fixed population size, and need a way of going from (parents + offspring) to next generation
- Often Deterministic
  - Fitness Based (rank all and select)
  - Age Based (prefer offspring)
  - Elitism (best n always live)

Initialization and Termination

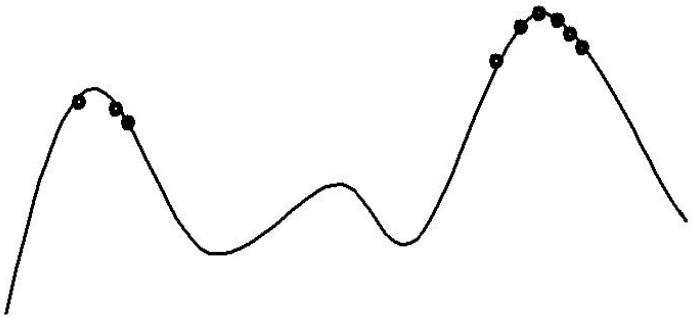
- Initialization usually done randomly
  - Need to ensure even mixture of candidates
  - Can include existing solutions, heuristics
- Termination Condition
  - Reach some known / desired fitness
  - Reach a generation limit
  - Reach a minimum diversity
  - Reach a generation limit of no improvement

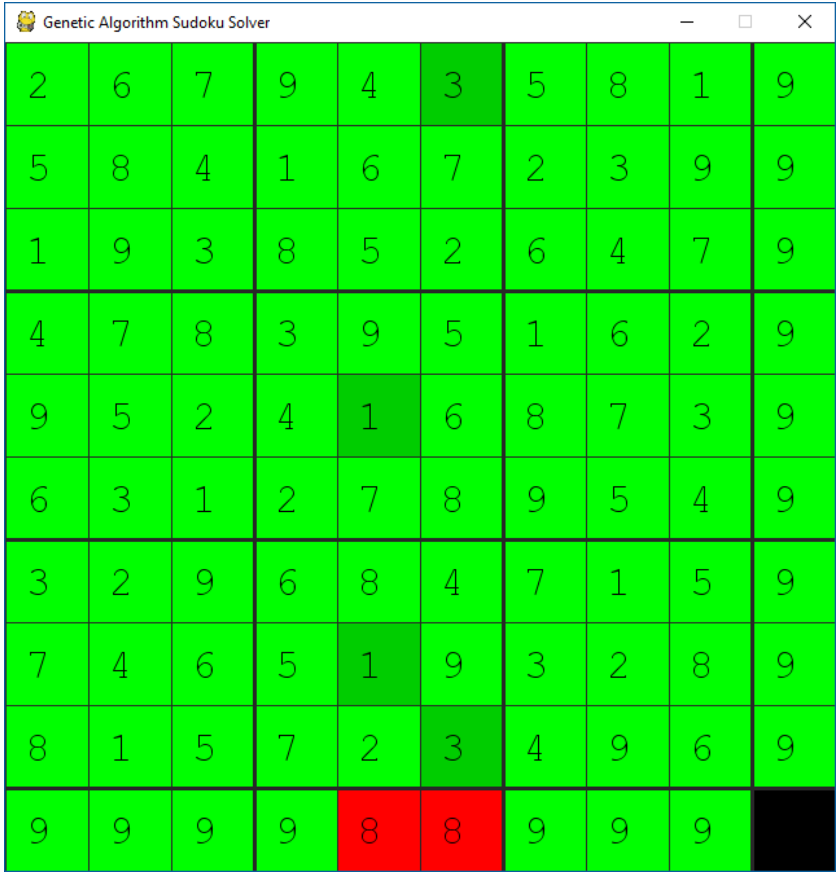

Population
GA Operators
Mutation
Crossover
Reproduction
Evaluation
Fitness value
Evolution Environment

Typical Behaviour of an EA

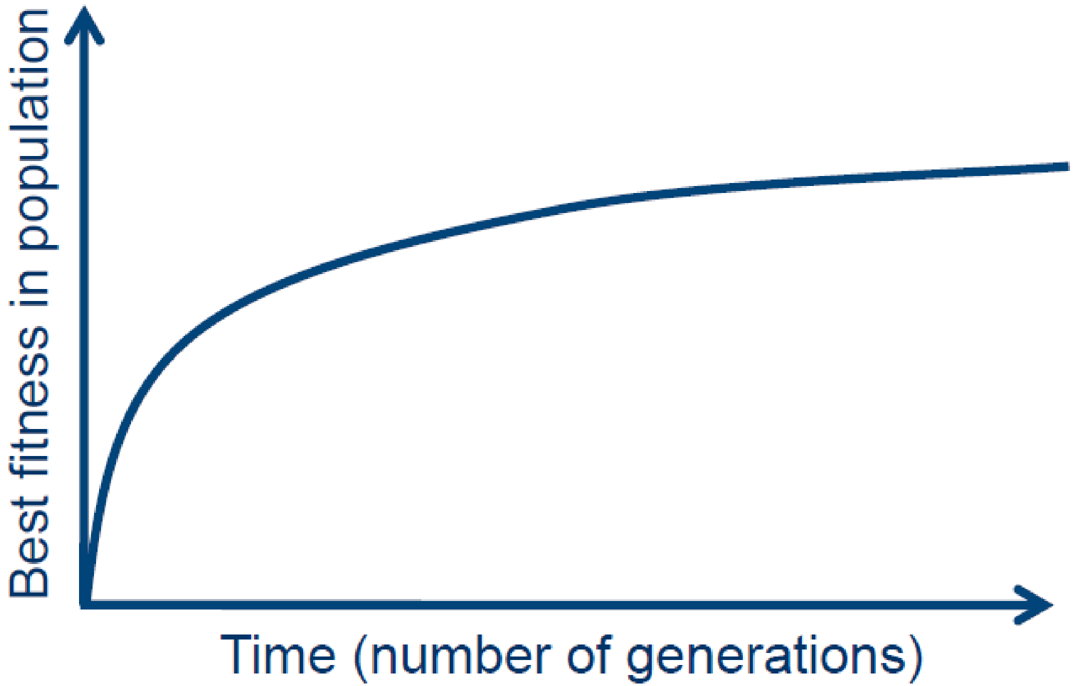


Local Maxima

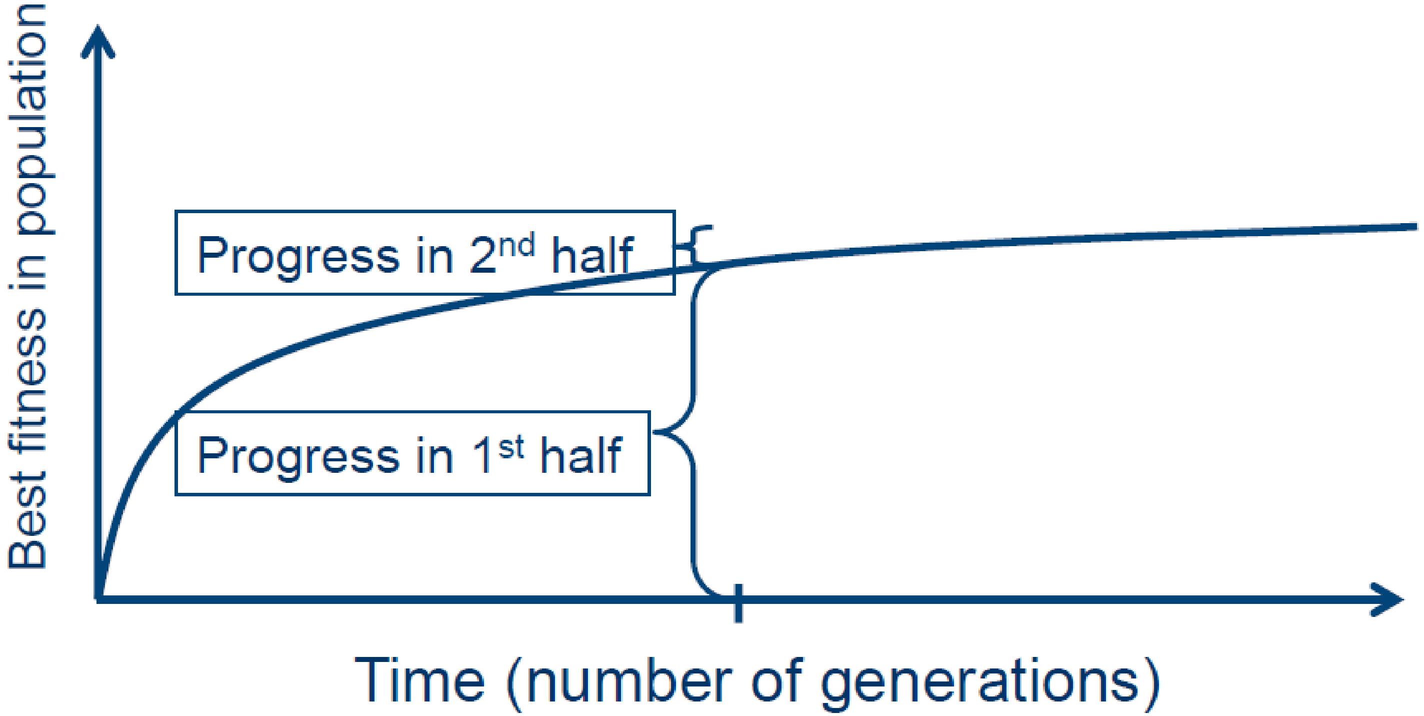




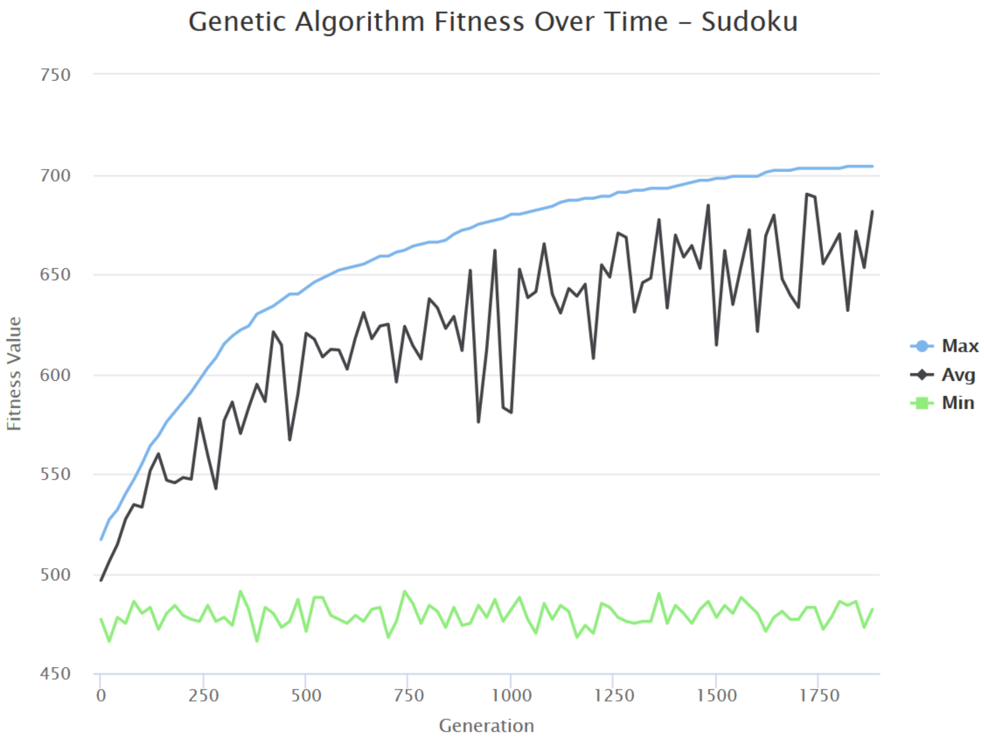
Typical GA Run



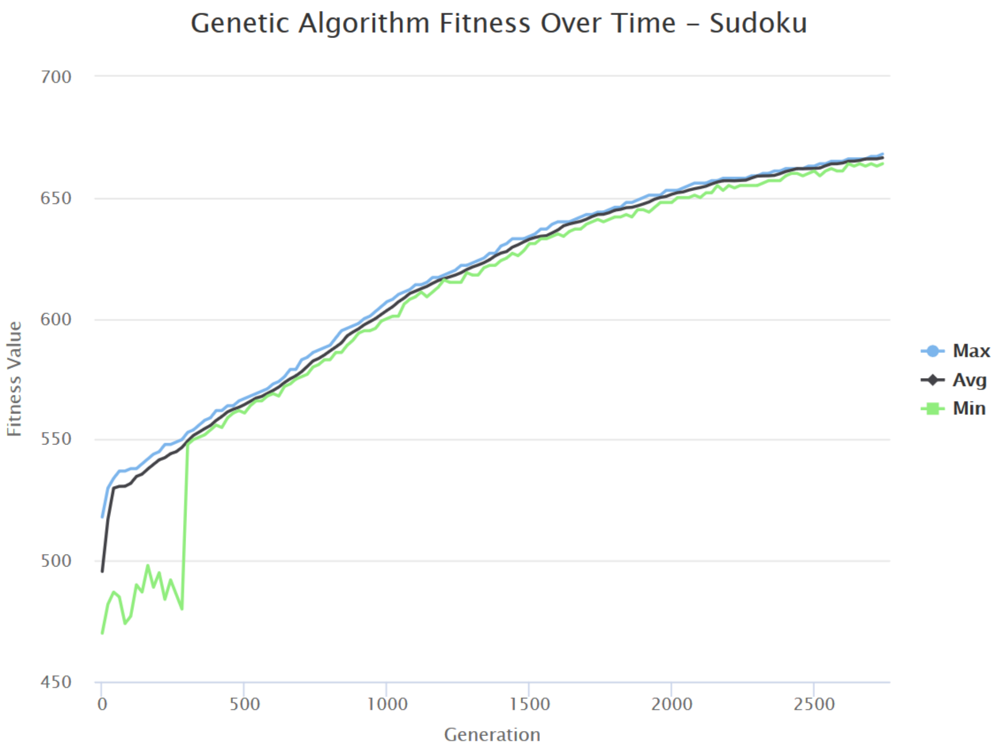
Running Time vs. Fitness



Population Diversity



Population Diversity



Evolutionary Algorithms

1. INITIALIZE population w/ random individuals
2. REPEAT UNTIL (termination condition)
3. EVALUTE population / individual fitness
4. SELECT parents with high fitness
5. COMBINE parents to form offspring
6. MUTATE resulting offspring
7. NEXT POP = select from [pop,offspring,parents]

Different Types of EA

- Different EA have different representations
  - Binary strings (Integers): Genetic Algorithms
  - Real-valued vectors: Evolution Strategies
  - Finite state machines: Evolutionary Programming
  - Tree Structure: Genetic Programming
- Differences largely cosmetic, best to
  - Choose representation to suit problem
  - Choose variation operators to suit representation

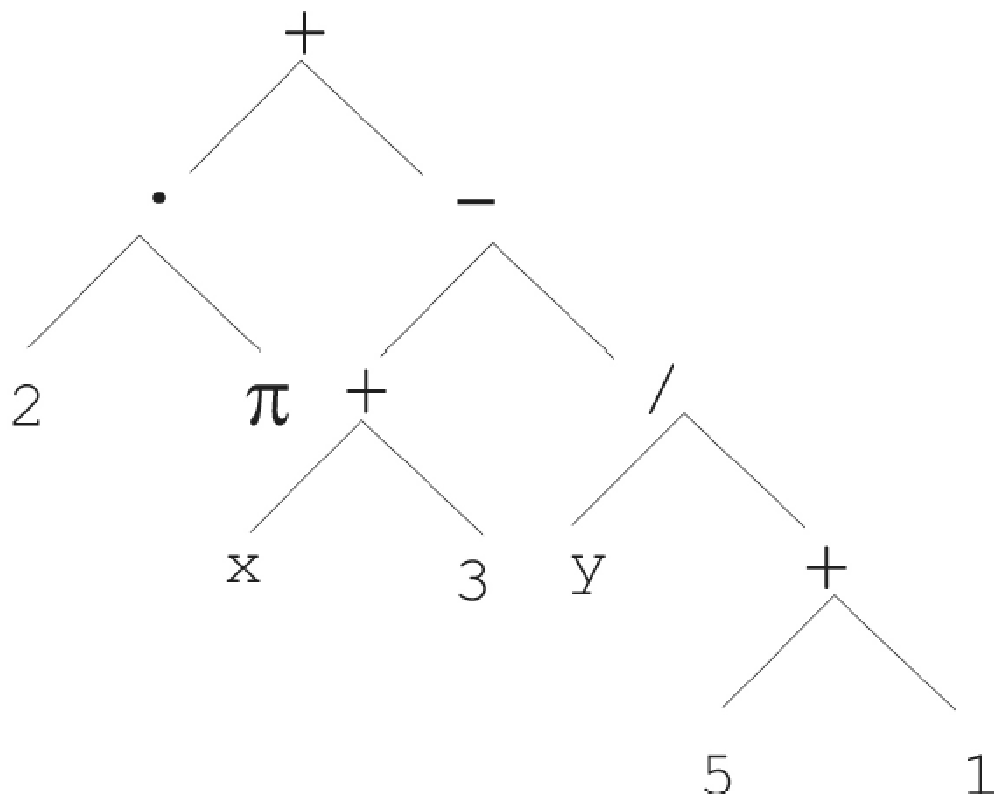
Genetic Programming

- Genotype as trees

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

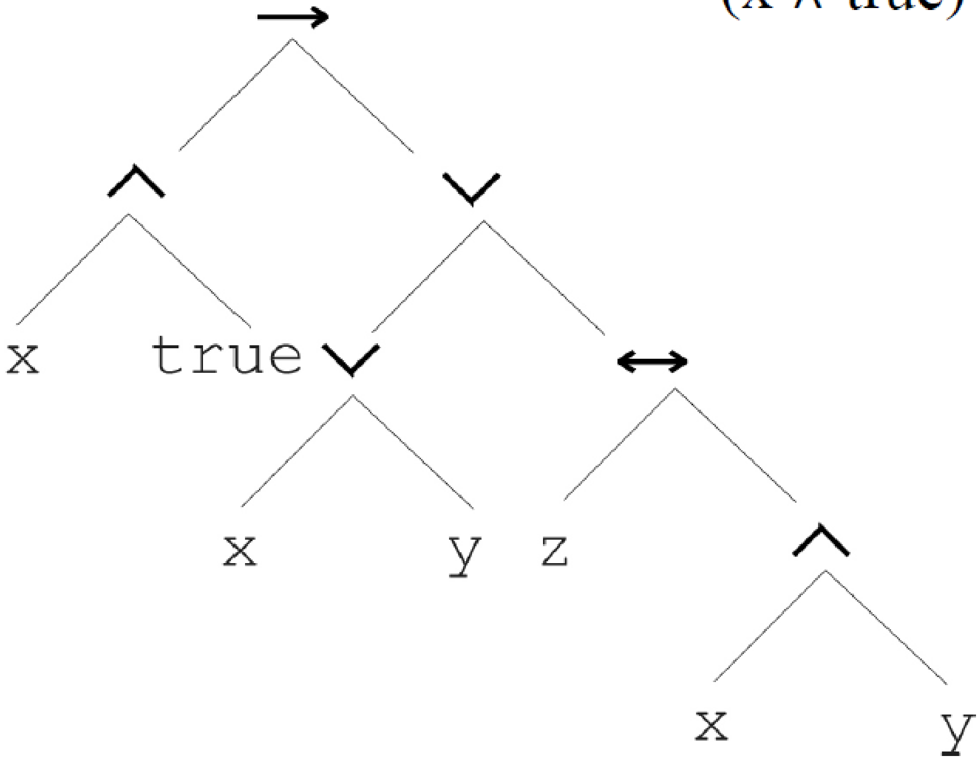
```
i =1;
while (i < 20)
{
        i = i +1
}
```

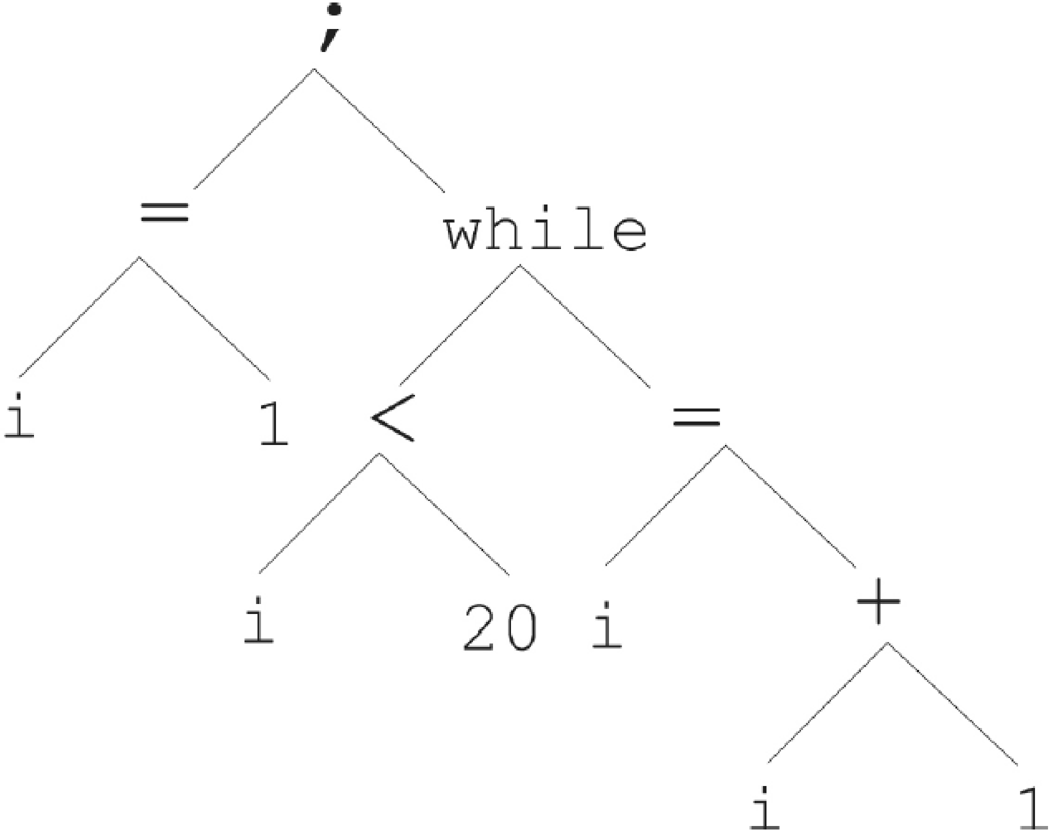
$$(x \wedge \text{true}) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$$



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

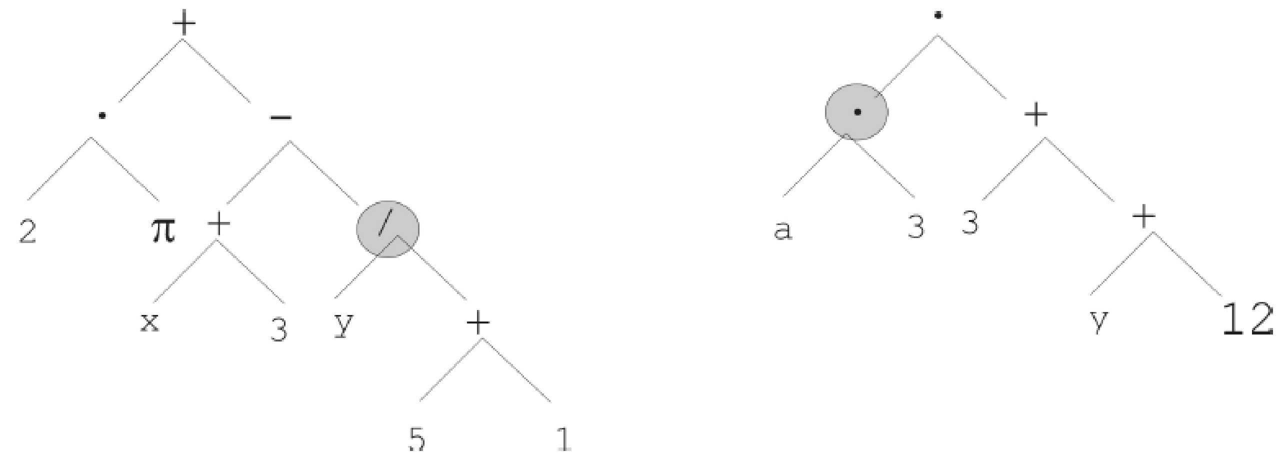
$$(x \wedge \text{true}) \rightarrow ((\, x \vee y \,) \vee (z \leftrightarrow (x \wedge y)))$$






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

parents



offspring

Genetic Algorithm Example

“Genetic Algorithm 2D Car Thingy”

https://rednuht.org/genetic_cars_2/