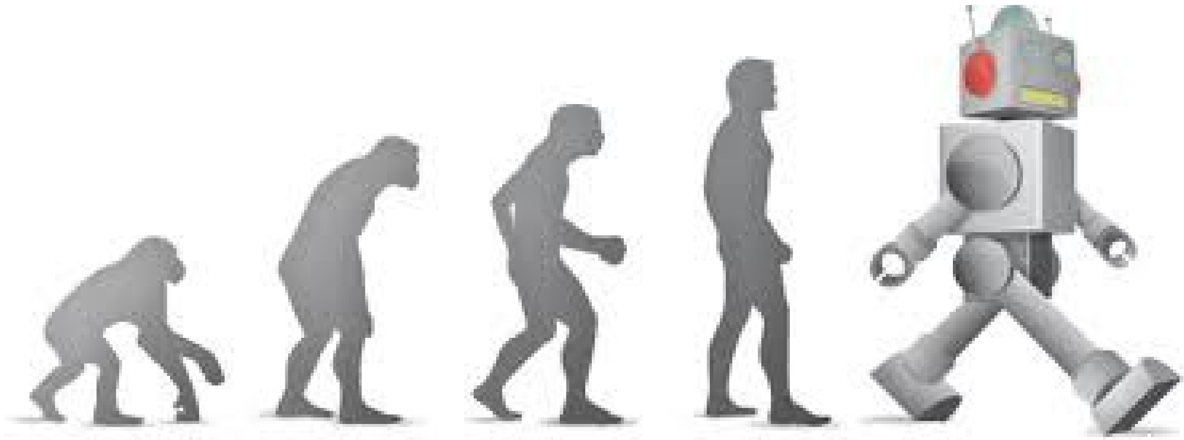
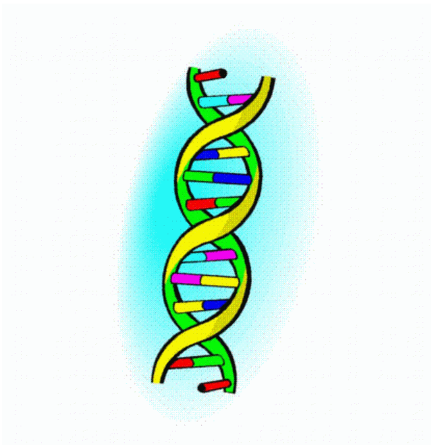
Genetic Algorithms

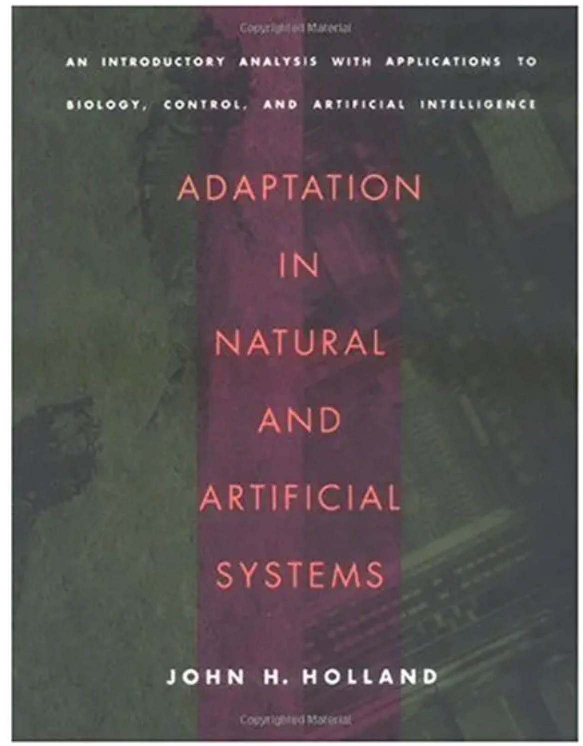




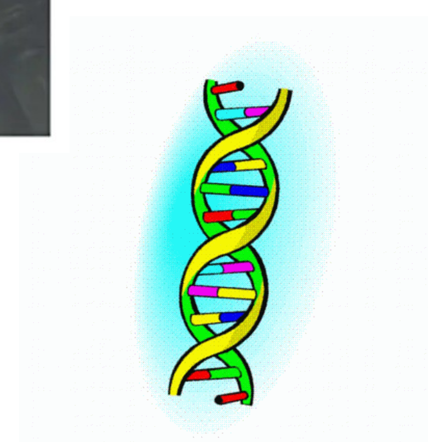
Genetic Algorithms

- Invented by John Holland in the 1960s

1929 - 2015

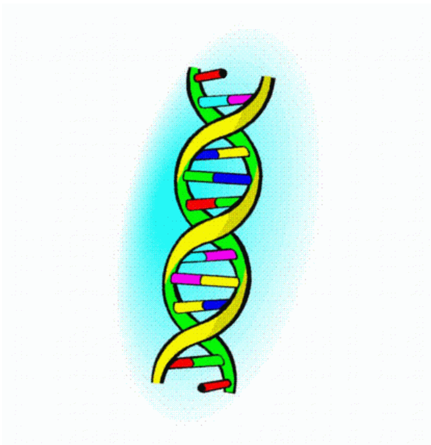


1975



Genetic Algorithms

- **Genome**
  - An encoded representation of a candidate solution to the problem (typically a sequence of numbers or bits)
- **Population of genomes**
  - A pool of candidate solutions, initially generated at random
- **Fitness function**
  - $f(genome) \Rightarrow$ numerical estimate of observed quality
- **Operators**
  - *Selection:* survival of the fittest
  - *Crossover:* genetic recombination
  - *Mutation:* random variation



Outline of a Genetic Algorithm

1. Create a new population of random genomes

000011101001100

110010011110011

100110100110100

00000100100010

111101011111101

101001101010101

000010000101011

110100100010001

Outline of a Genetic Algorithm

1. Create a new population of random genomes

2. Evaluate the fitness of each genome in the population

$f($000011101001100$) = 6$

$f($110010011110011$) = 9$

$f($100110100110100$) = 7$

$f($111101011111101$) = 12$

$f($000000100100010$) = 3$

$f($101001101010101$) = 8$

$f($000010000101011$) = 5$

$f($110100100010001$) = 6$

Outline of a Genetic Algorithm

1. Create a new population of random genomes

2. Evaluate the fitness of each genome in the population

3. Build a new generation of genomes:

   (a) **select** 2 genomes probabilistically, based on fitness

$f($000011101001100$) = 6$

$f($110010011110011$) = 9$

$f($100110100110100$) = 7$

$f($111101011111101$) = 12$

$f($00000100100010$) = 3$

$f($101001101010101$) = 8$

$f($000010000101011$) = 5$

$f($110100100010001$) = 6$

Outline of a Genetic Algorithm

1. Create a new population of random genomes

2. Evaluate the fitness of each genome in the population

3. Build a new generation of genomes:

   (a) **select** 2 genomes probabilistically, based on fitness

10011010011010 0

11110101111110 1

Outline of a Genetic Algorithm

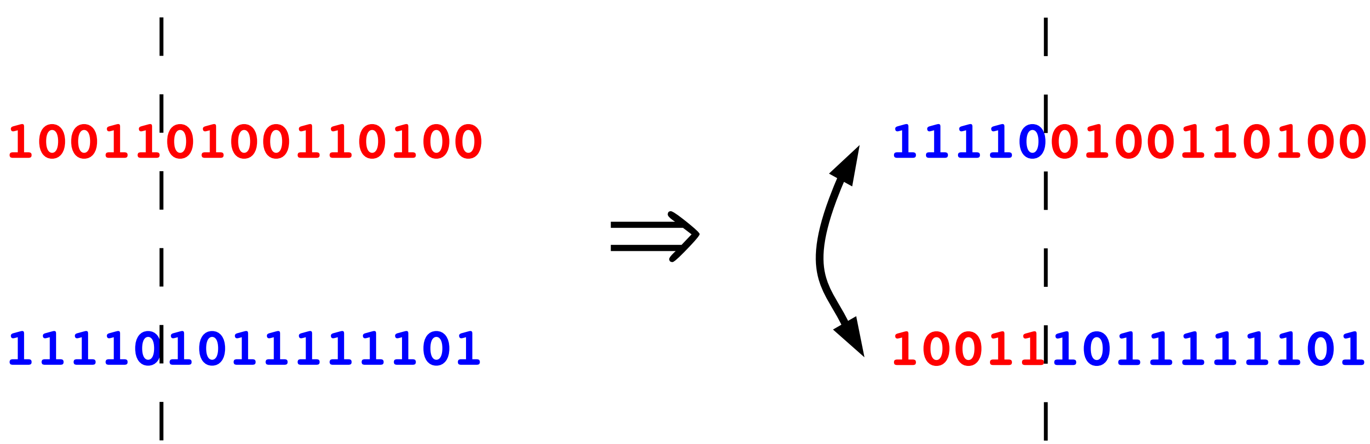
1. Create a new population of random genomes

2. Evaluate the fitness of each genome in the population

3. Build a new generation of genomes:

    (a) **select** 2 genomes probabilistically, based on fitness

    (b) create 2 new offspring from them, using **crossover**



Outline of a Genetic Algorithm

1. Create a new population of random genomes

2. Evaluate the fitness of each genome in the population

3. Build a new generation of genomes:

    (a) **select** 2 genomes probabilistically, based on fitness

    (b) create 2 new offspring from them, using **crossover**

111100100110100

100111011111101

Outline of a Genetic Algorithm

1. Create a new population of random genomes

2. Evaluate the fitness of each genome in the population

3. Build a new generation of genomes:

   (a) **select** 2 genomes probabilistically, based on fitness

   (b) create 2 new offspring from them, using **crossover**

   (c) **mutate** each offspring with some small probability

111100000110110

000111011111101

Outline of a Genetic Algorithm

1. Create a new population of random genomes

2. Evaluate the fitness of each genome in the population

3. Build a new generation of genomes:

    (a) **select** 2 genomes probabilistically, based on fitness

    (b) create 2 new offspring from them, using **crossover**

    (c) **mutate** each offspring with some small probability

    (d) add the offspring to the new generation

111100000110110

000111011111101

Outline of a Genetic Algorithm

1. Create a new population of random genomes

2. Evaluate the fitness of each genome in the population

3. Build a new generation of genomes:

   (a) **select** 2 genomes probabilistically, based on fitness

   (b) create 2 new offspring from them, using **crossover**

   (c) **mutate** each offspring with some small probability

   (d) add the offspring to the new generation

   … continue steps (a) - (d)

Outline of a Genetic Algorithm

1. Create a new population of random genomes

2. Evaluate the fitness of each genome in the population

3. Build a new generation of genomes:

   (a) **select** 2 genomes probabilistically, based on fitness

   (b) create 2 new offspring from them, using **crossover**

   (c) **mutate** each offspring with some small probability

   (d) add the offspring to the new generation

4. When the new generation has reached the same size as the current population, replace the current population by the new generation … and repeat

Outline of a Genetic Algorithm

- Over time, the **average fitness** of the population will increase
- The best-fit individuals are not guaranteed to survive to the next generation
- Even the worst-fit individuals have some (small) probability of surviving
- Some GAs use **elitism** to ensure that the best individuals do survive
- Many ways of probabilistically selecting individuals
  - fitness-proportionate selection (“roulette-wheel sampling”)
  - rank selection
  - tournament selection

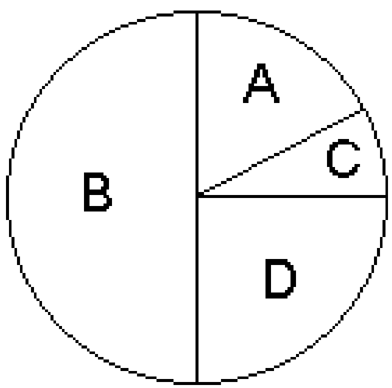
A Simple Example

	Genome	Fitness	New population
A:	**00000110**	2	
B:	**11101110**	6	
C:	**00100000**	1	
D:	**00110100**	3	

Average fitness of current population = 12 / 4 = 3.0

A Simple Example

	Genome	Fitness	New population
A:	**00000110**	2	
B:	**11101110**	6	
C:	**00100000**	1	
D:	**00110100**	3	



Fitness-proportionate selection
("roulette-wheel sampling")

B: **11101110** and C: **00100000** are selected

A Simple Example

Genome	Fitness	New population
A: **00000110**	2	
B: **11101110**	6	
C: **00100000**	1	
D: **00110100**	3	

No crossover

B: **11101110**

C: **00100000**

A Simple Example

	Genome	Fitness	New population
A:	**00000110**	2	
B:	**11101110**	6	
C:	**00100000**	1	
D:	**00110100**	3	

No crossover

B is mutated

B: **11101110** → E: **01101110**

C: **00100000**

A Simple Example

Genome	Fitness	New population
A: **00000110**	2	E: **01101110**
B: **11101110**	6	C: **00100000**
C: **00100000**	1	
D: **00110100**	3	

No crossover

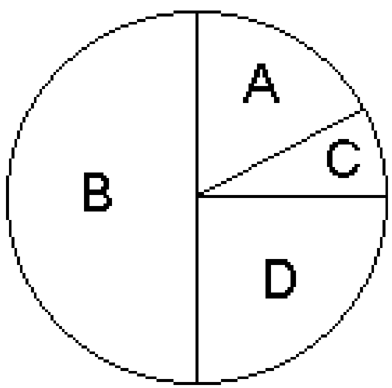
B is mutated

B: **11101110** → E: **01101110**

C: **00100000**

A Simple Example

Genome	Fitness	New population
A: **00000110**	2	E: **01101110**
B: **11101110**	6	C: **00100000**
C: **00100000**	1	
D: **00110100**	3	



Fitness-proportionate selection
("roulette-wheel sampling")

B: **11101110** and D: **00110100** are selected

A Simple Example

Genome	Fitness	New population
A: **00000110**	2	E: **01101110**
B: **11101110**	6	C: **00100000**
C: **00100000**	1	
D: **00110100**	3	

Crossover occurs

B: **11101110** → G: **01101110**

D: **00110100** → F: **10110100**

A Simple Example

Genome	Fitness	New population
A: **00000110**	2	E: **01101110**
B: **11101110**	6	C: **00100000**
C: **00100000**	1	
D: **00110100**	3	

	Crossover occurs	F is mutated
B: **11101110**	F: **10110100** →	H: **10110000**
D: **00110100**	G: **01101110**	

A Simple Example

Genome	Fitness	New population
A: **00000110**	2	E: **01101110**
B: **11101110**	6	C: **00100000**
C: **00100000**	1	H: **10110000**
D: **00110100**	3	G: **01101110**

Crossover occurs F is mutated

B: **11101110**

D: **00110100**

F: **10110100** → H: **10110000**

G: **01101110**

A Simple Example

Genome	Fitness	New population	Fitness
A: **00000110**	2	E: **01101110**	5
B: **11101110**	6	C: **00100000**	1
C: **00100000**	1	H: **10110000**	3
D: **00110100**	3	G: **01101110**	5

Average fitness of new population = 14 / 4 = 3.5

Best-fit genome from previous population was lost

Demo: Evolving an English Phrase

- Genome
  a sequence of letters representing an English phrase
- Fitness function
  the number of letters in the genome that match
  "the rain in spain stays mainly in the plain"
- Example
  "the yain in szbin stays mainly ik the ploin"
  fitness = 38
- GA parameters
  population size: 100
  crossover probability: 0.75
  mutation probability: 0.005 per locus

Key Concepts

- Search space
- Fitness landscape
- Local minimum / maximum
- Hill-climbing search
- Population-based search