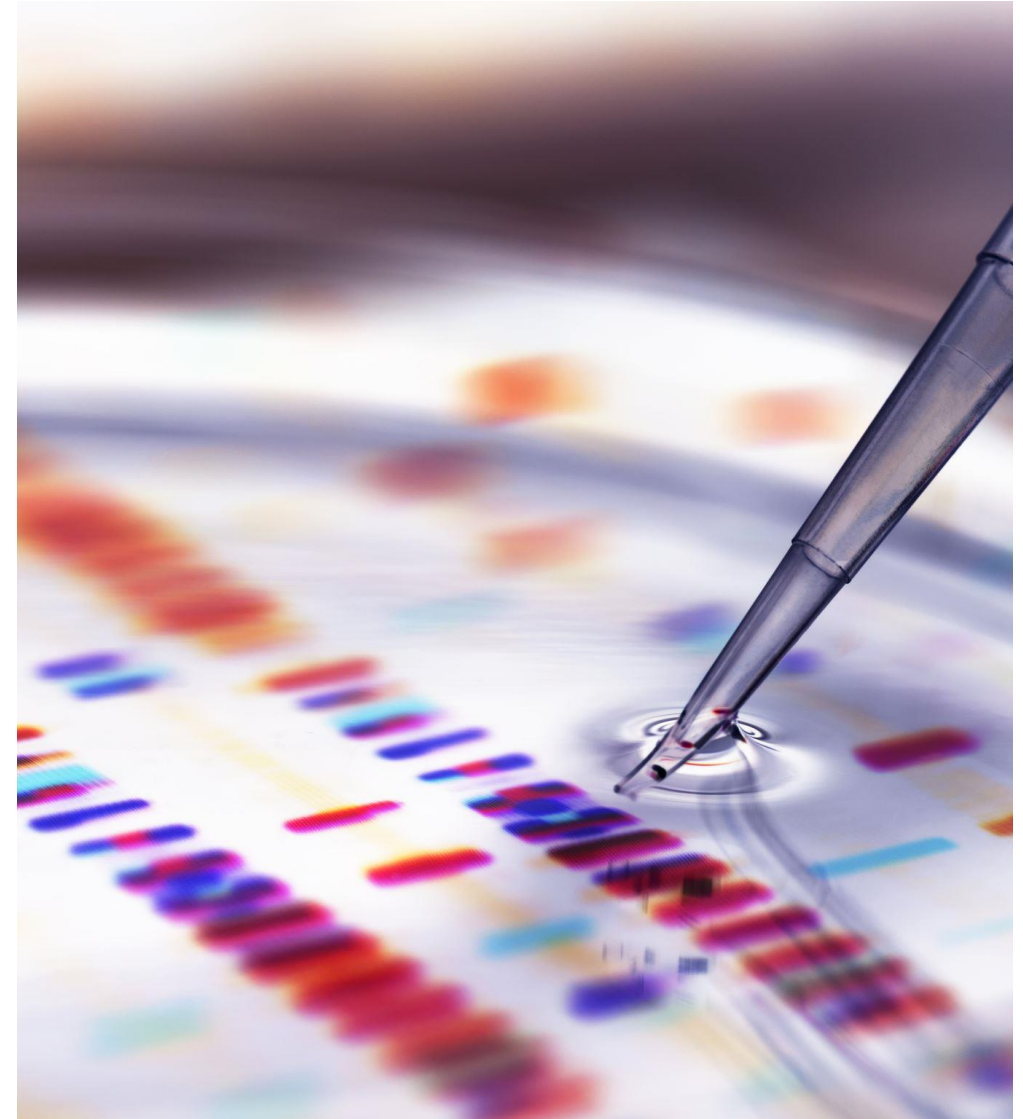Alaa Sheta

Genetic Algorithms

Outlines

- What are genetic algorithms?
- How do they work?
- Representation
- Selection
- Reproduction
- Fitness evaluation
- Application of GAs

Definition

"Genetic Algorithms are good at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, solutions you might not otherwise find in a lifetime."

– Salvatore Mangano
Computer Design, May 1995

What are GAs?

- Genetic algorithms (GAs) are powerful computing search techniques that can find approximate solutions to various optimization and search problems.

- Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology, such as inheritance, mutation, selection, and crossover (also called recombination). Genetic algorithms are categorized as global search heuristics.
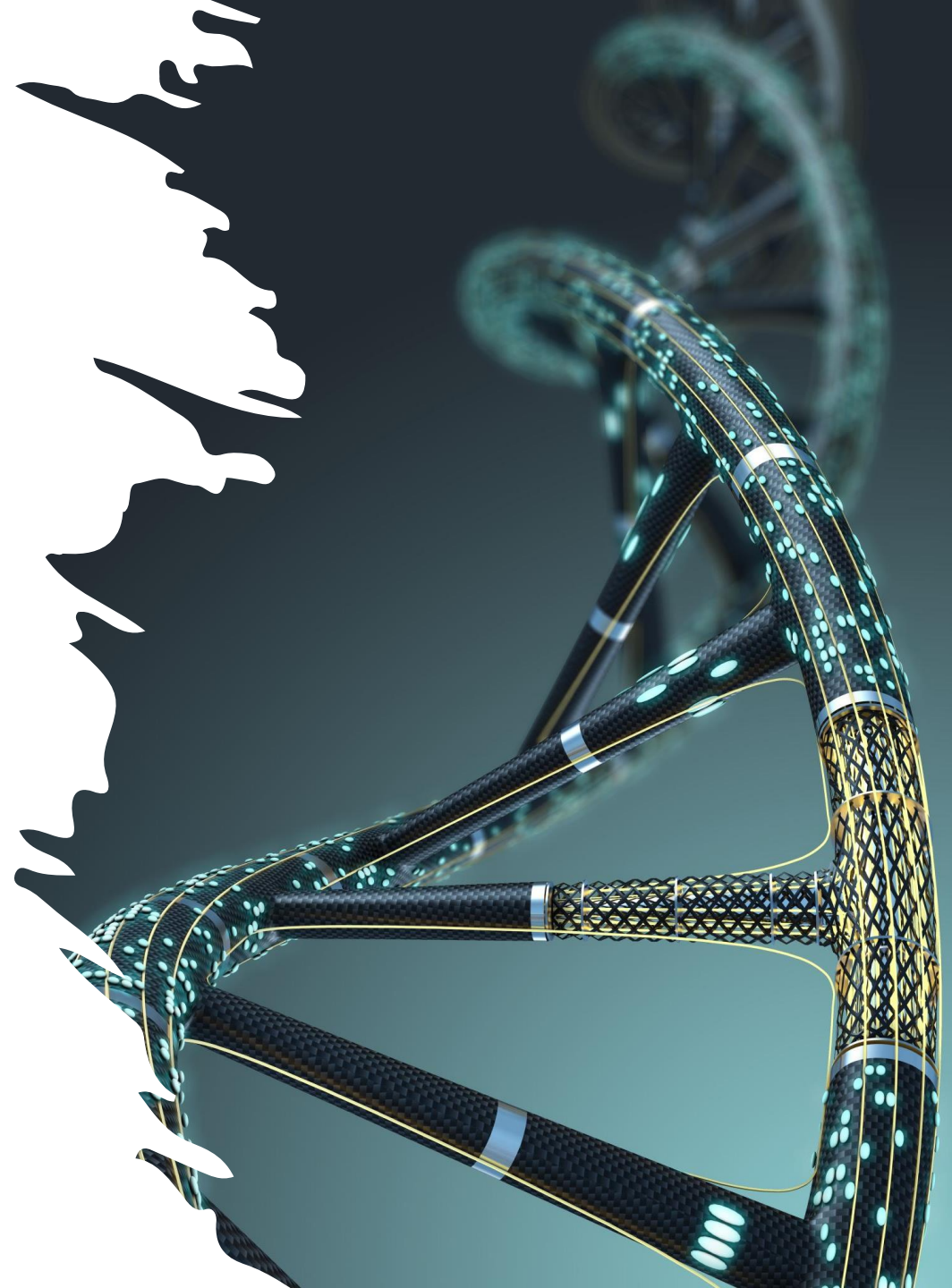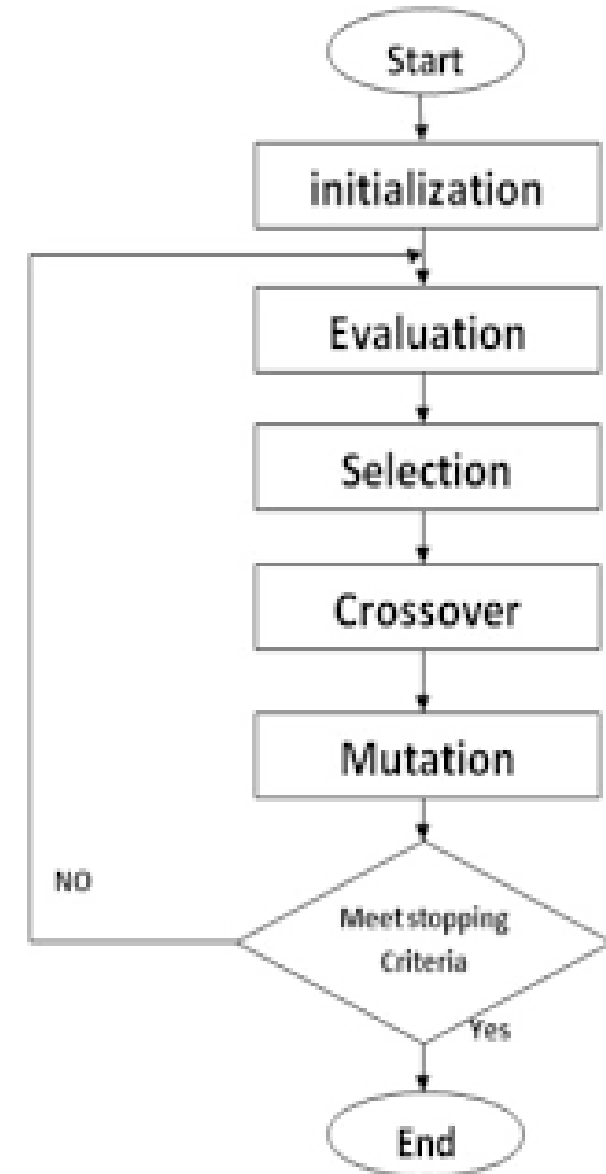
What are GAs?

- Genetic algorithms are implemented as computer simulations in which a population of abstract representations (called chromosomes, the genotype, or the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions.

- Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible (Holland 1975).
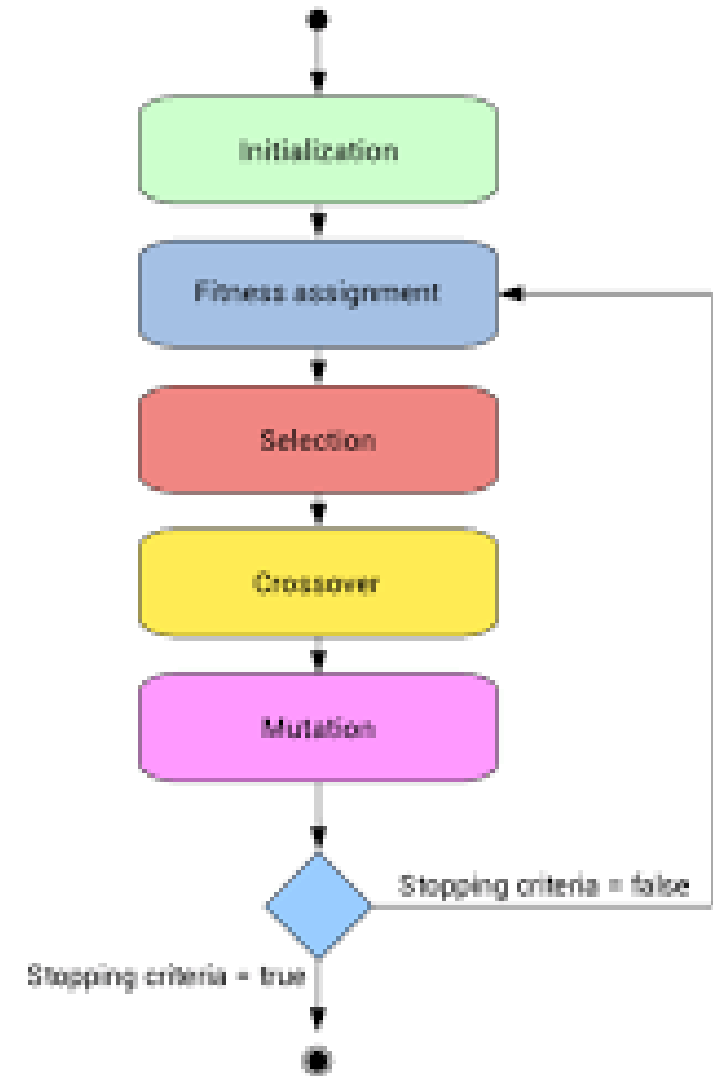
What are GAs?

- The evolution usually starts from a population of randomly generated individuals and continues in generations.

- In each generation, the fitness of every individual in the population is evaluated, and multiple individuals are selected from the current population (based on their fitness) and modified (recombined and possibly mutated) to form a new population.

What are GAs?

- The new population is then used in the next iteration of the algorithm.

- Commonly, the algorithm terminates when a maximum number of generations has been produced or a satisfactory fitness level has been reached for the population.

- If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

Binary GAs

Algorithm 1 Genetic Algorithm

1: determine objective function (OF)
2: assign number of generation to 0 (t=0)
3: randomly create individuals in initial population P(t)
4: evaluate individuals in population P(t) using OF
5: **while** termination criterion is not satisfied **do**
6:     t=t+1
7:     select the individuals to population P(t) from P(t-1)
8:     change individuals of P(t) using crossover and mutation
9:     evaluate individuals in population P(t) using OF
10: **end while**
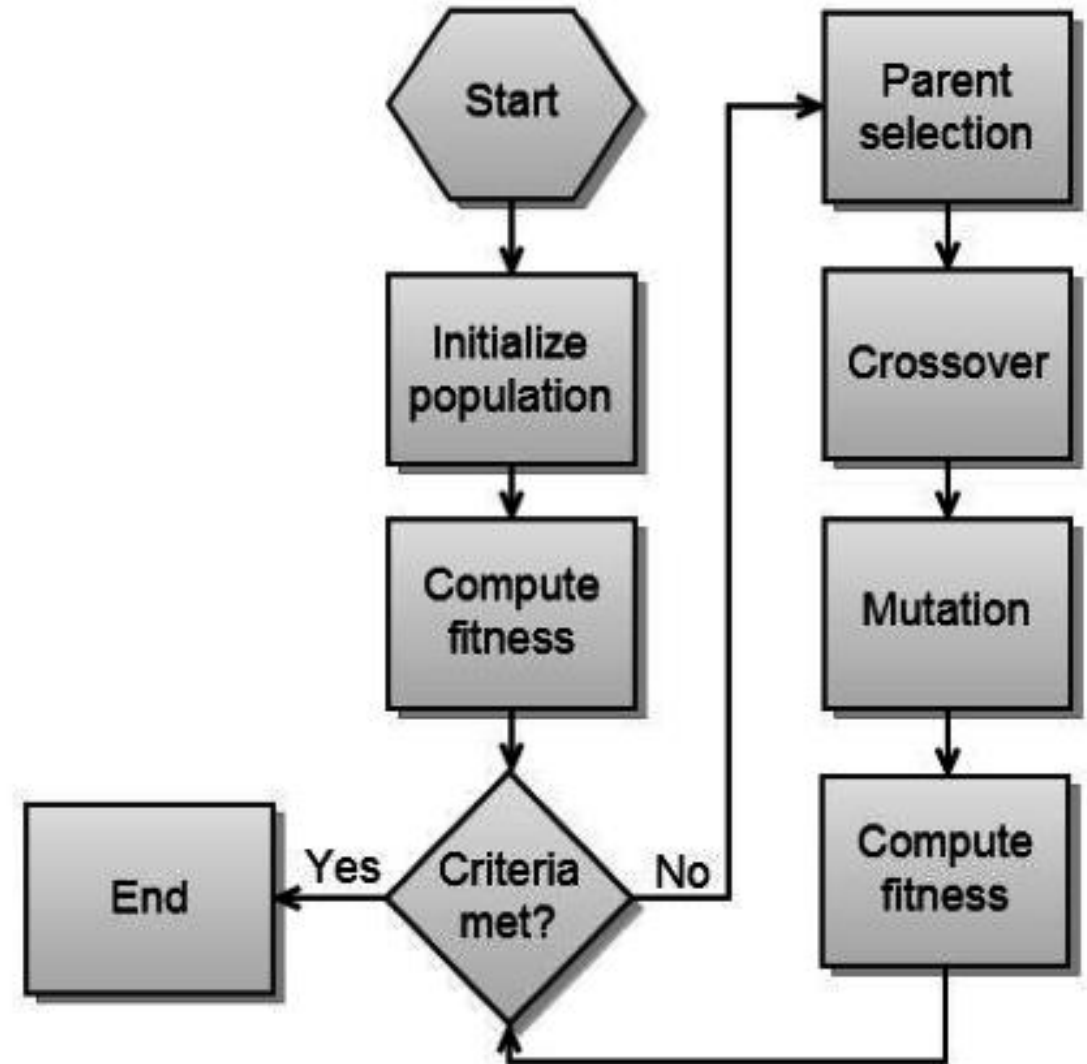11: return the best individual found during the evolution

Key terms

- **Individual** – Any possible solution
- **Population** – Group of all *individuals*
- **Search Space** – All possible solutions to the problem
- **Chromosome** – Blueprint for an *individual*
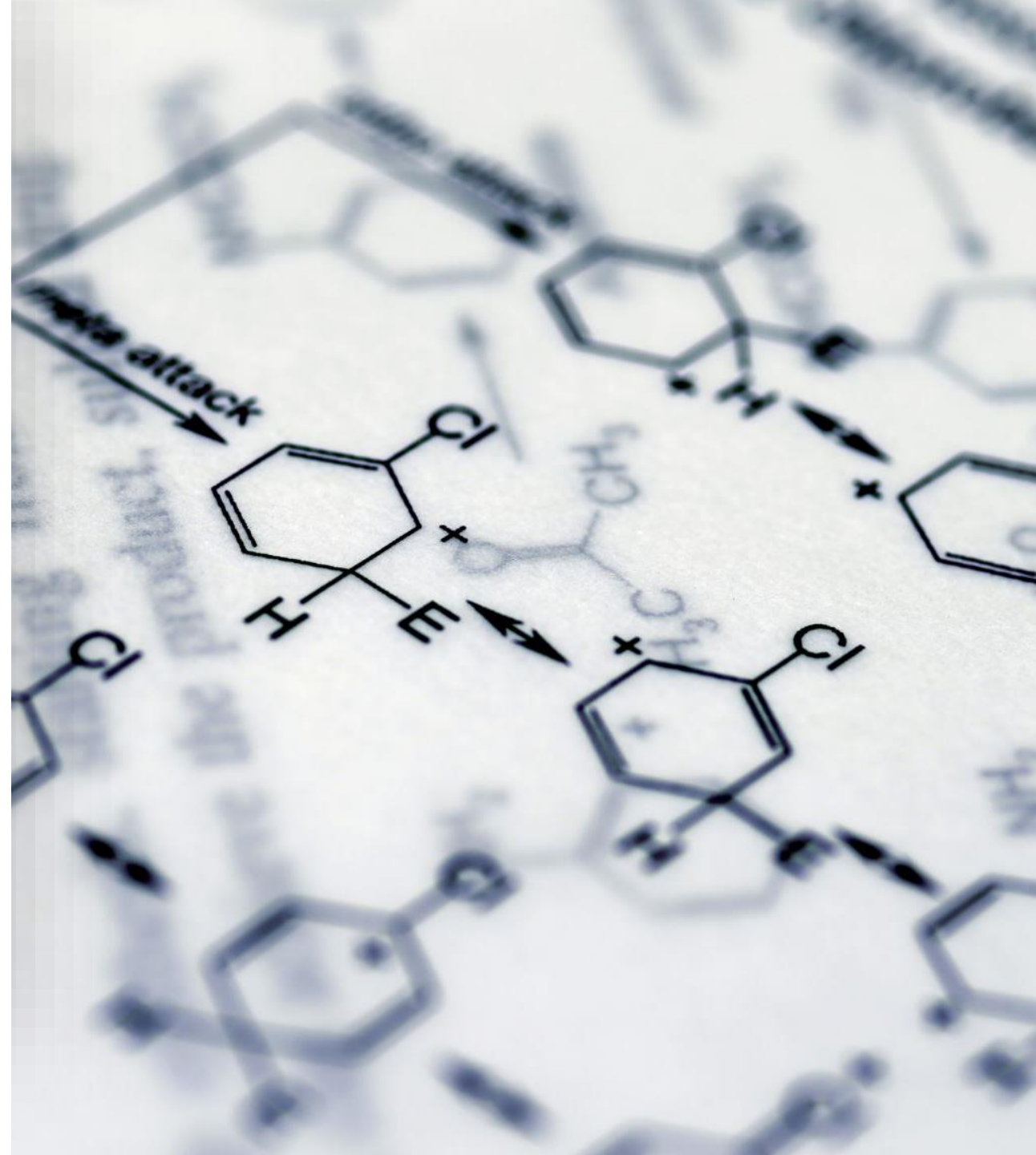
GA Workflow

- Diagram of GA workflow: Initialization →
  Selection → Crossover → Mutation →
  Fitness Evaluation → Termination.

Initialization

- Methods to initialize the population:
  - Random initialization
  - Heuristic-based initialization (optional)
- Importance of diversity in the initial population.

random   evenly scattered   knowledge-based
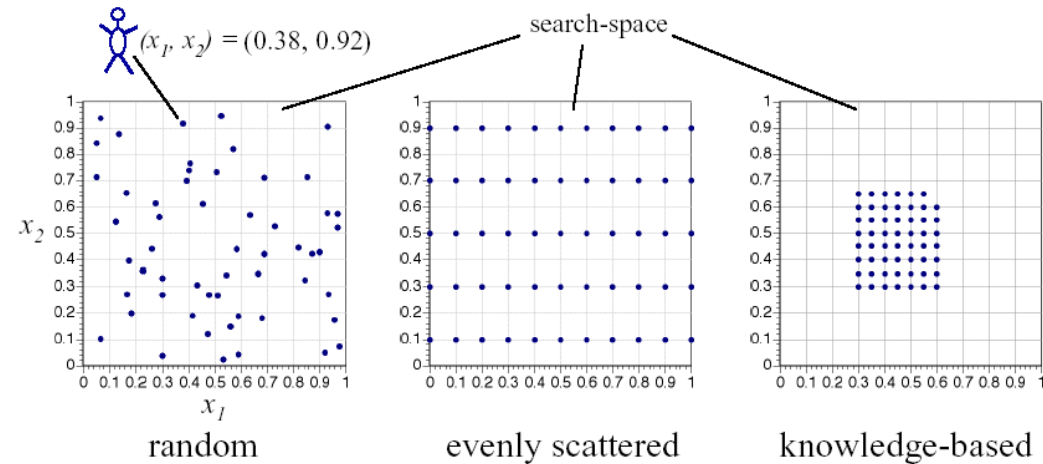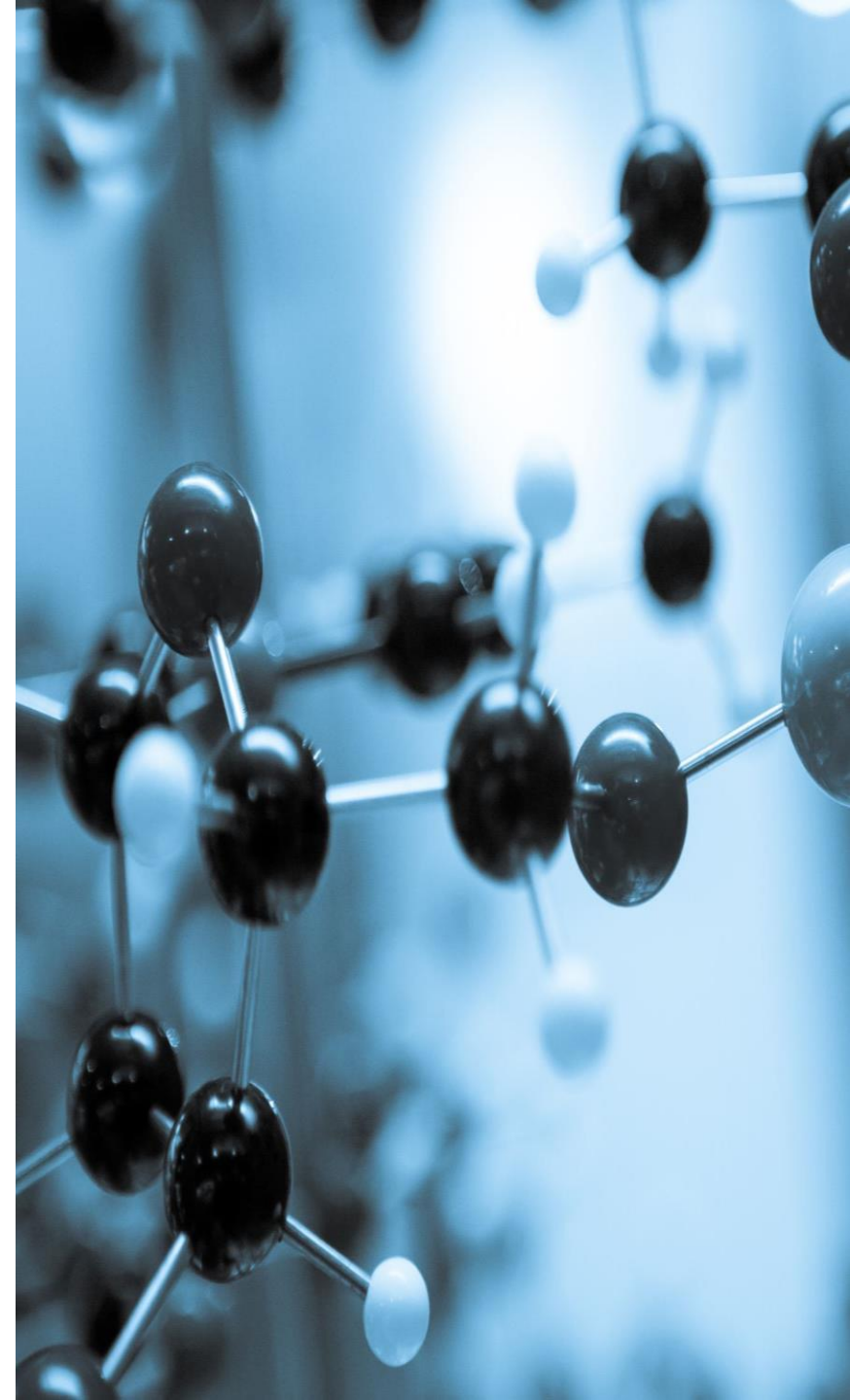
Population Diversity

- One of the most critical factors determining the genetic algorithm's performance is population diversity.

- If the average distance between individuals is large, the diversity is high; if the average distance is small, the diversity is low.

- Getting the proper diversity is a matter of trial and error.

- The genetic algorithm might not perform well if the diversity is too high or too low.

Population diversity

Diversity in the initial population of a Genetic Algorithm (GA) is critical for several reasons:

- **Exploration of the Solution Space**: Diversity ensures a broad coverage of the solution space, which is essential for effective search.

- Avoiding Premature Convergence: Premature convergence occurs when the population becomes uniform too early, often leading the GA to settle on suboptimal solutions, known as local optima.

- A diverse initial population reduces the likelihood of premature convergence by providing various starting points, **allowing the algorithm to explore different paths and avoid getting trapped in poor solutions.**

Population diversity

- **Balancing Exploration and Exploitation**:
  - Genetic algorithms rely on balancing exploration (searching new areas of the solution space) and exploitation (refining known good areas).
  - A diverse initial population emphasizes exploration, giving the GA the flexibility to search broadly before narrowing its focus on refining high-quality solutions through selection and reproduction processes.
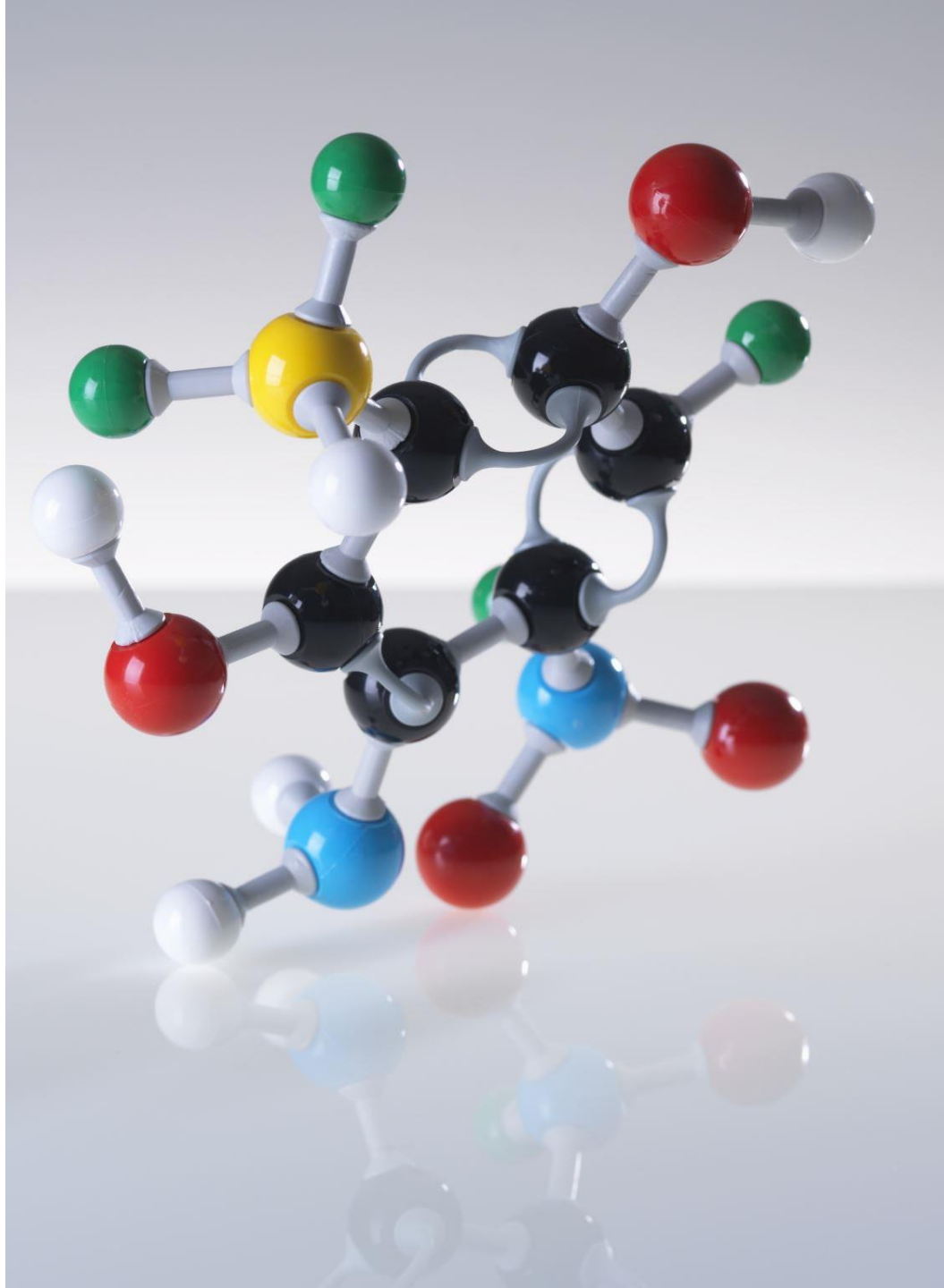
Selection Mechanism
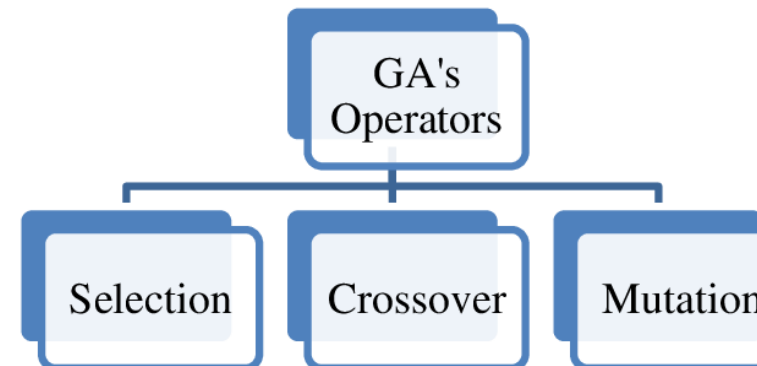
- Selection is a critical phase in GAs, determining which individuals will contribute to the next generation. Individuals with higher fitness scores, indicating their suitability as potential solutions, are more likely to be selected.

- The goal is to simulate the survival of the fittest, favoring solutions that perform well in the given problem domain.

- Popular selection methods:
    - **Roulette Wheel Selection:** Probability of selection proportional to fitness.
    - **Tournament Selection:** A random subset of the population, the fittest individual selected.
    - **Rank Selection:** Individuals ranked, with a probability of selection based on rank.

Roulette-wheel selection

- Roulette-wheel selection, also known as fitness-proportionate selection, is used in genetic algorithms to select individuals (solutions) from a population for reproduction based on their fitness.

- The idea is that individuals with higher fitness are more likely to be selected, but even less fit individuals have a chance, which maintains genetic diversity.
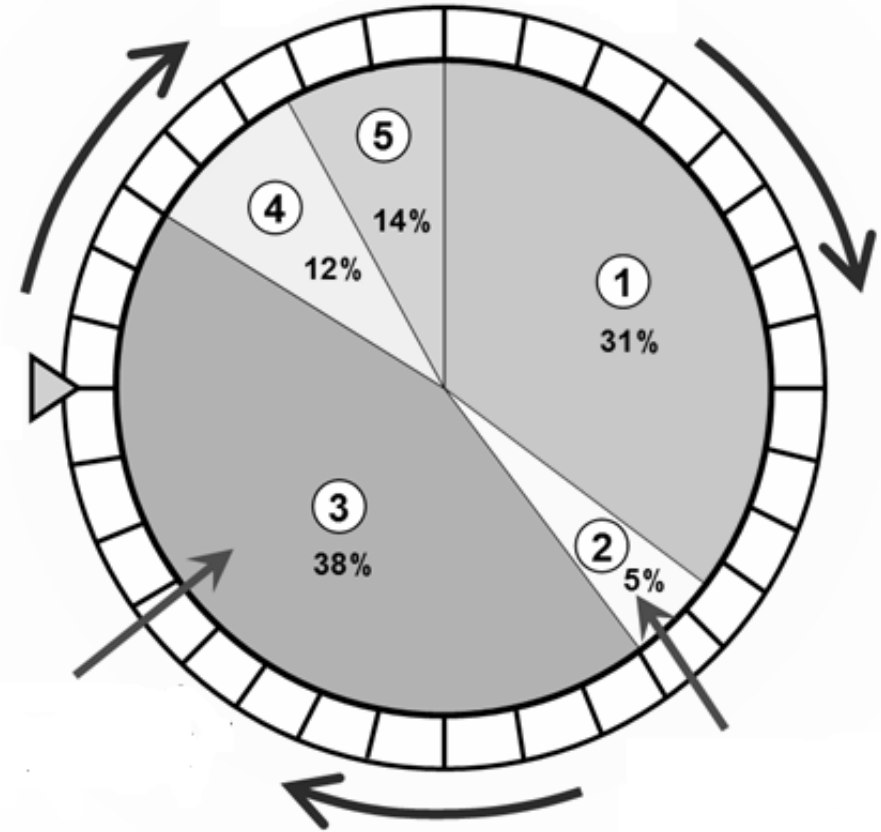
Example of Roulette-Wheel Selection

- Suppose a population has four individuals with fitness values of 10, 20, 30, and 40. The total fitness is 10 + 20 + 30 + 40 = 100. Everyone's probability of selection is calculated.

- With these probabilities, individuals with higher fitness (such as Individual 4) are more likely to be selected but do not eliminate the chances of less-fit individuals, allowing for some diversity in selection.

- Individual 1: $\frac{10}{100} = 0.1$ or 10%

- Individual 2: $\frac{20}{100} = 0.2$ or 20%

- Individual 3: $\frac{30}{100} = 0.3$ or 30%

- Individual 4: $\frac{40}{100} = 0.4$ or 40%

Roulette Wheel Selection

	String $\mathbf{s}_i$	Fitness $f(\mathbf{s}_i)$	Relative Fitness
$\mathbf{s}_1$	10110	2.23	0.14
$\mathbf{s}_2$	11000	7.27	0.47
$\mathbf{s}_3$	11110	1.05	0.07
$\mathbf{s}_4$	01001	3.35	0.21
$\mathbf{s}_5$	00110	1.69	0.11
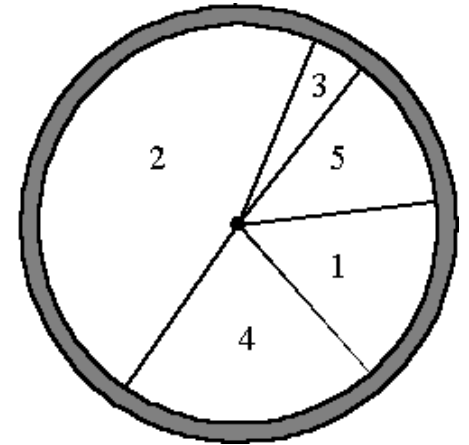
Number of individual	1	2	3	4	5	6	7	8	9	10	11
fitness value	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.0
selection probability	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02	0.0

Selection probability and fitness value

- The above table shows the selection probability for 11 individuals.

- Individual 1 is the most fit individual and occupies the largest interval, whereas individual 10 as the second least fit individual has the smallest interval on the line (see figure).

- Individual 11, the least fit interval, has a fitness value of 0 and get no chance for reproduction
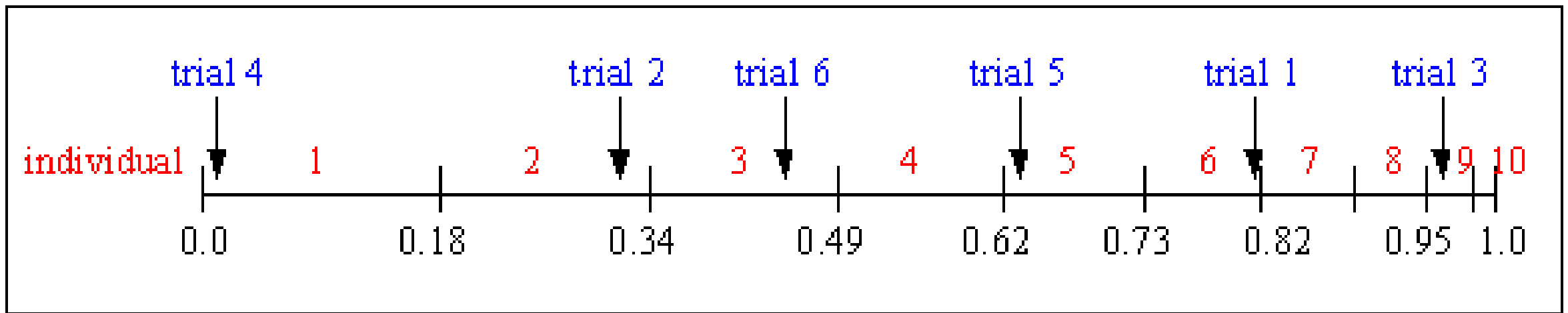
Example

- To select the mating population, the appropriate number of uniformly distributed random numbers (uniformly distributed between 0.0 and 1.0) is independently generated.
  - Sample of 6 random numbers: 0.81, 0.32, 0.96, 0.01, 0.65, 0.42.
  - After selection, the mating population consists of individuals 1, 2, 3, 5, 6, and 9.
- The roulette-wheel selection algorithm provides a zero bias but does not guarantee minimum spread.

Disadvantages of Roulette-wheel selection

- Because selection is directly proportional to fitness, it is possible that strong individuals may dominate in producing offspring, thereby limiting the diversity of the new population.

- In other words, Roulette-wheel (i.e., proportional selection) has a high selective pressure.

| Selection Method	Advantages	Limitations
Roulette-Wheel Selection	- Rewards high-fitness individuals proportionally - Intuitive method closely tied to fitness values - Encourages exploitation of the best solutions	- Sensitive to high fitness variability; may cause premature convergence - Low-fit individuals have minimal chance of selection, reducing diversity

Stochastic universal sampling

- Stochastic Universal Sampling (SUS) is a selection technique commonly used in genetic algorithms to select individuals from a population proportionate to their fitness values.

- This method is designed to maintain diversity and ensure that selection pressure doesn't become too high.

- It allows even low-fitness individuals a chance to be selected, which can prevent premature convergence.

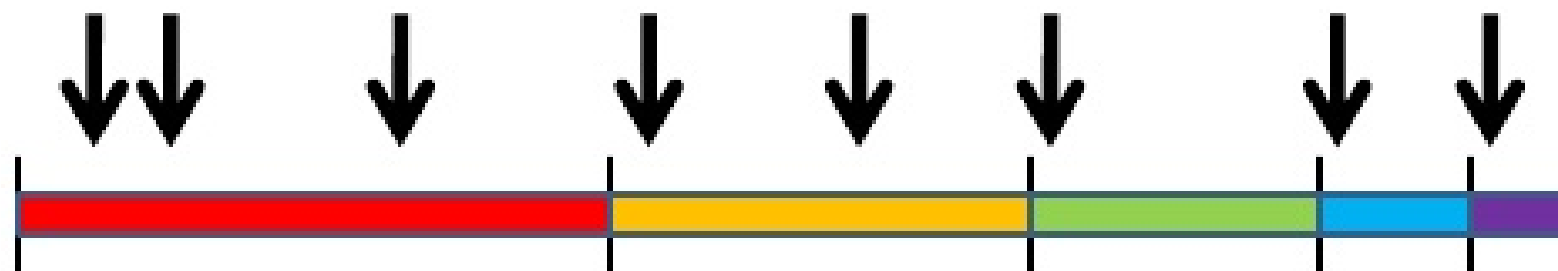# Key Concepts of Stochastic Universal Sampling

1. **Proportional Selection**: Like other fitness-proportionate selection methods (e.g., roulette wheel selection), SUS selects individuals based on their fitness. The probability of an individual being selected is proportional to its fitness compared to the population.

2. **Spread of Pointers**: Instead of selecting individuals one by one, SUS uses a single "pointer" that is moved in fixed increments. This way, multiple individuals can be chosen in a single pass, allowing for a more diverse selection.

3. **Deterministic yet Fair**: SUS avoids some of the randomness seen in standard roulette wheel selection by providing a fairer and more uniform distribution of selection across individuals. This helps maintain a stable gene pool and avoids heavily favoring high-fitness individuals, potentially leading to a loss of genetic diversity.
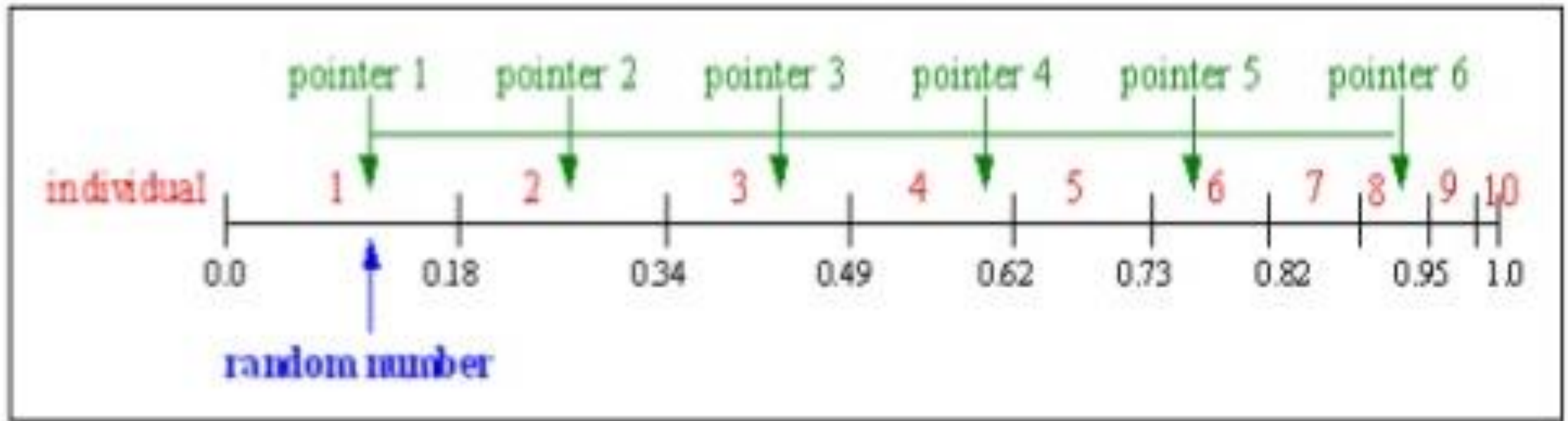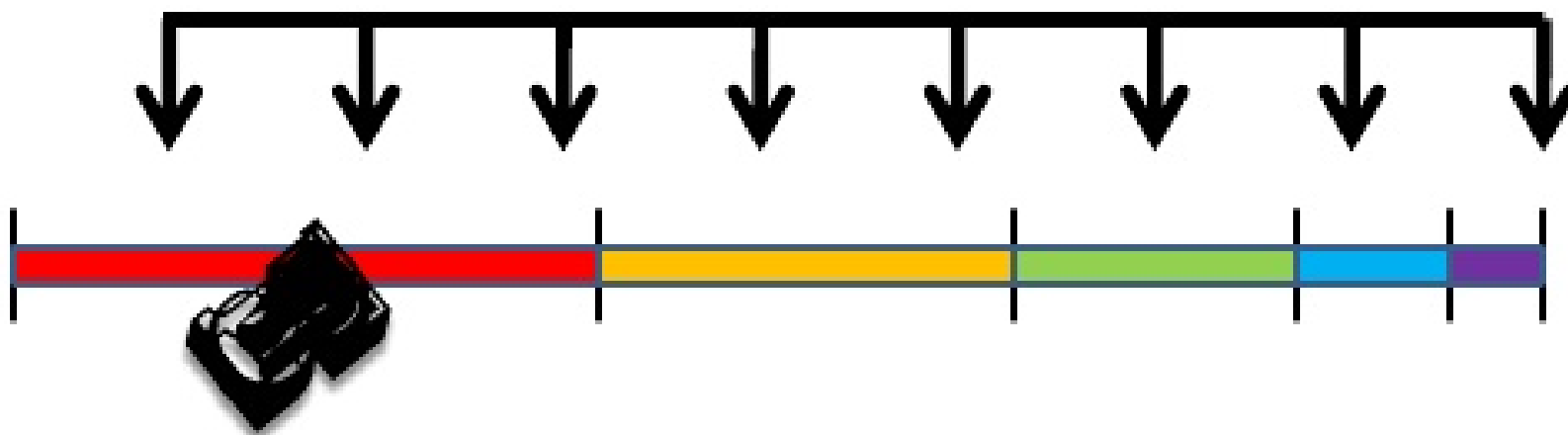
Stochastic universal sampling

- Unlike RWS, making n selection only takes a single span.

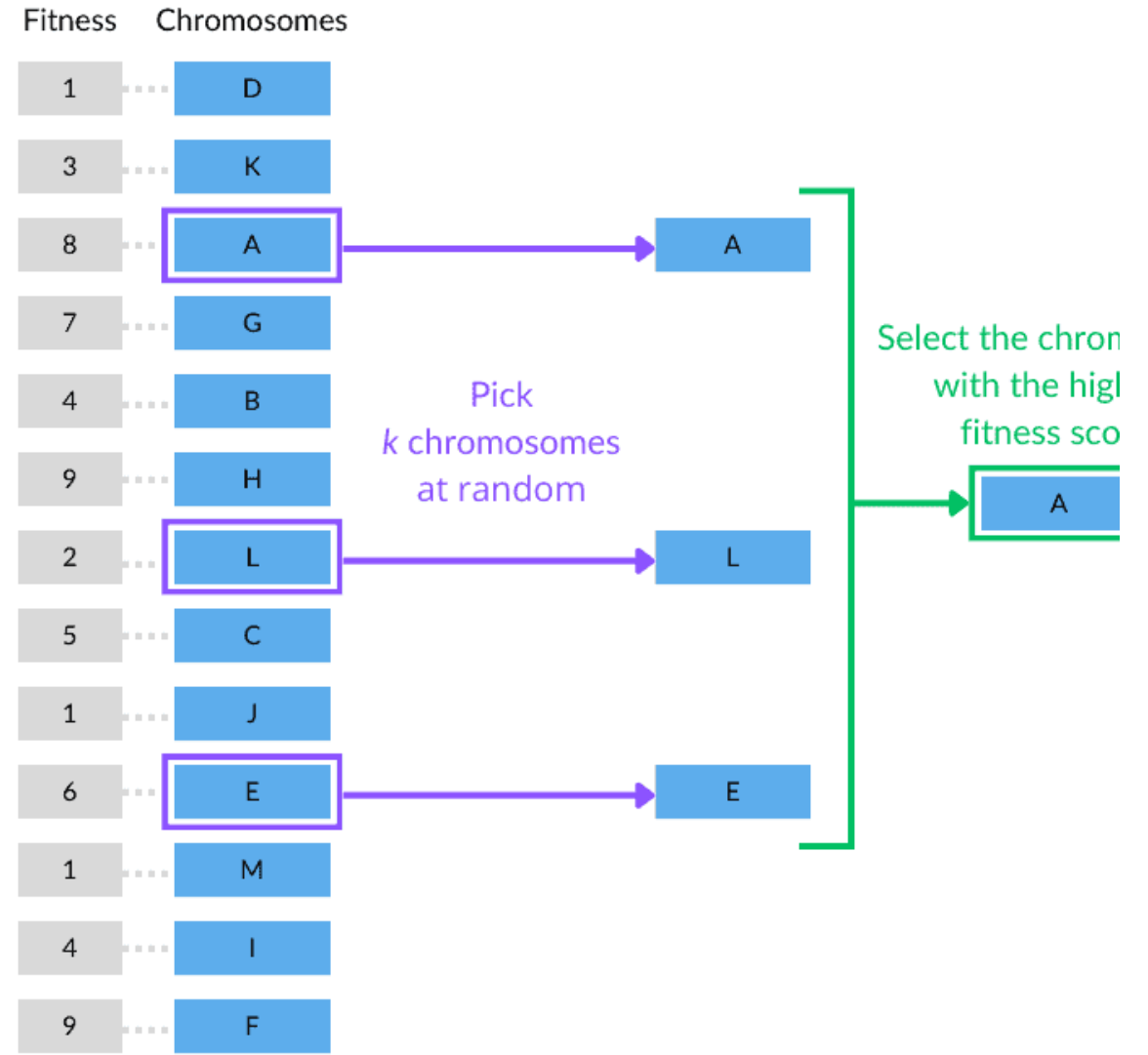- The selection process proceeds by advancing around the wheel in equal-sized steps.

Tournament Selection

- Tournament selection is a popular method for selecting individuals from a population for reproduction.

- In this case, the tournament size is set to 3, i.e., k = 3. Thus, we randomly picked chromosomes A, L, and E from the population. Soon, we compared the fitness of the three chromosomes and selected A, which had the highest score.

- After that, we just need to run this tournament "n" time to select the complete set of individuals who will be the next generation's parents.

Selection Method	Advantages	Limitations
Tournament Selection	- Adjustable selection pressure by varying tournament size - Can work without calculating absolute fitness probabilities - Simple and computationally efficient	- Higher selection pressure with large tournament sizes may lead to loss of diversity - Premature convergence possible with small populations and high selection pressure

Rank Selection

- Rank Selection is a method used in genetic algorithms (GAs) that ranks individuals based on their fitness rather than directly using their absolute fitness values.

- This approach aims to reduce the risk of premature convergence and helps maintain a steady selection pressure even if there is a large difference between the best and worst fitness values.

Example of Rank Selection

Assume we have a population of 5 individuals. After ranking them by fitness, we assign ranks from 1 to 5:

- **Individual 1** (highest fitness) is ranked 5.

- **Individual 2** has rank 4.

- **Individual 3** has rank 3.

- **Individual 4** has rank 2.

- **Individual 5** (lowest fitness) is ranked 1.

Instead of selecting probabilities based on fitness values, we base them on rank. In a linear scaling setup, the highest-ranked individual could have a probability of 30%, with a steady reduction for each rank down.

Selection Method	Advantages	Limitations
Rank-Based Selection	- Maintains diversity by reducing dominance of high-fitness individuals - Stabilizes selection pressure regardless of fitness scale - Effective for populations with wide fitness range	- Slower convergence due to balanced selection pressure - Less precision in fitness-based selection; small fitness differences may be ignored

Key Steps in Reproduction

- In Genetic Algorithms (GAs), **reproduction** is generating new individuals (solutions) in each generation to form the next population.

- The main goal of reproduction is to create offspring that inherit characteristics from parent individuals, allowing the algorithm to explore and exploit the solution space to find optimal or near-optimal solutions.

- Reproduction in GAs typically involves two fundamental processes: **selection** and **genetic operators** (crossover and mutation).

Selection

Cross Over   Mutation

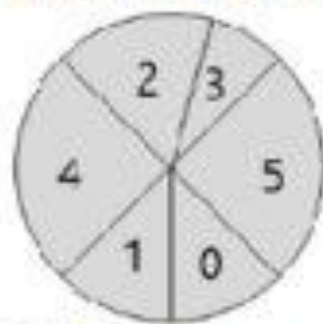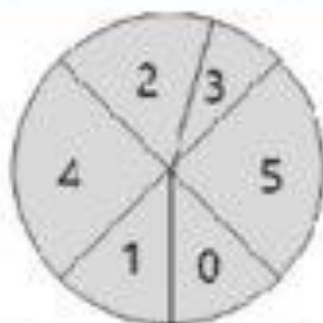Reproduction

Gene

Chromosome

Evaluation of Fitness

Population
(Generation 0)

Selection

Population
(Generation 1)

Next
Generation

Reproduction Operators

- Reproduction produces offspring from selected parents by applying crossover and/or mutation operators.
    - Crossover
    - Mutation
- These two leading operators help the evolutionary process converge to the optimal solution domain.
- For each operator, we define a certain probability of development.
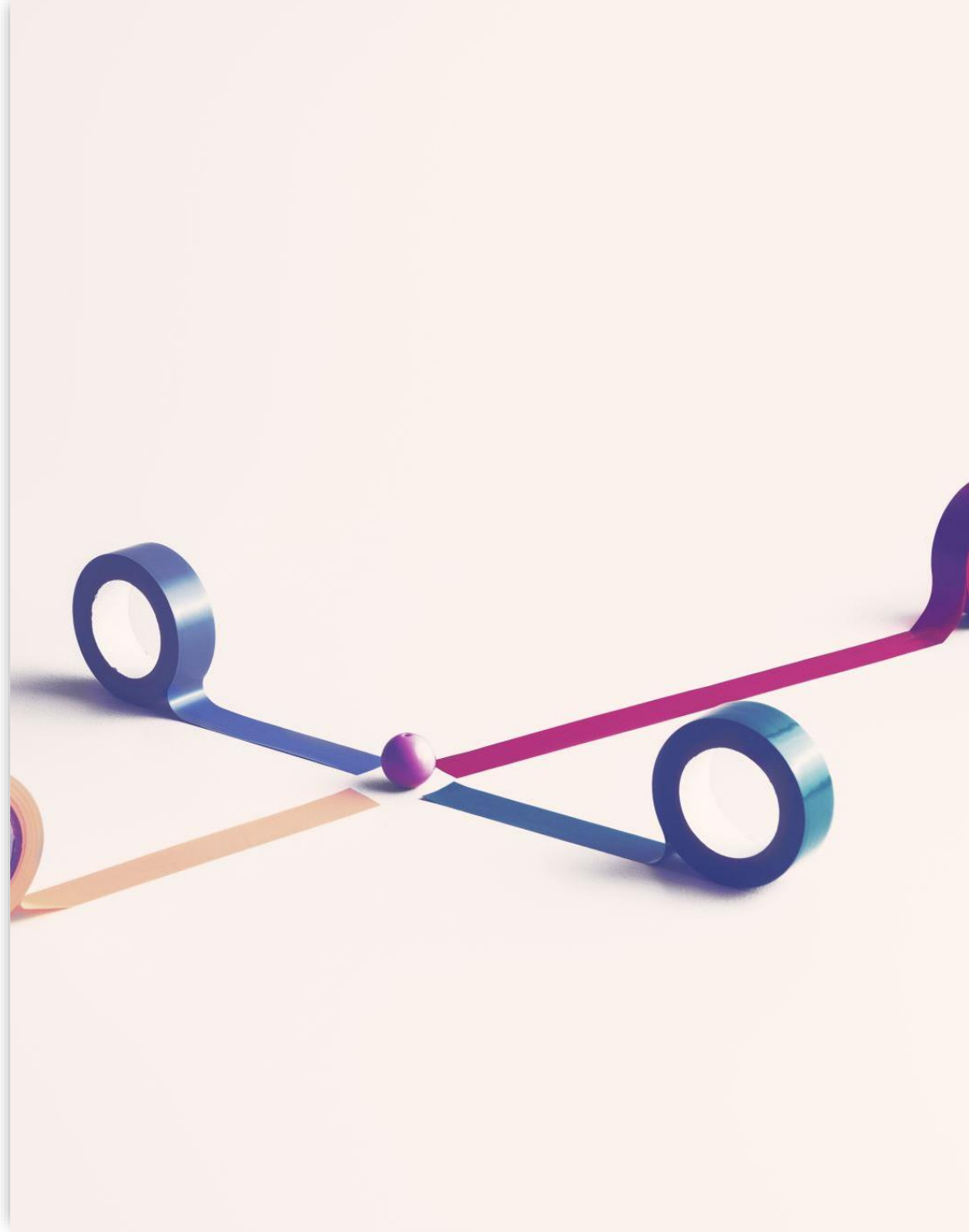
Crossover (Recombination)

- Explanation of crossover: Combining two parent chromosomes to produce offspring.

- Common crossover techniques:
    - **Single-point crossover:** One crossover point between parents.
    - **Two-point crossover:** Two crossover points between parents.
    - **Uniform crossover:** Genes are randomly exchanged between parents.

- Effect of crossover on exploration and exploitation.

Single-point crossover
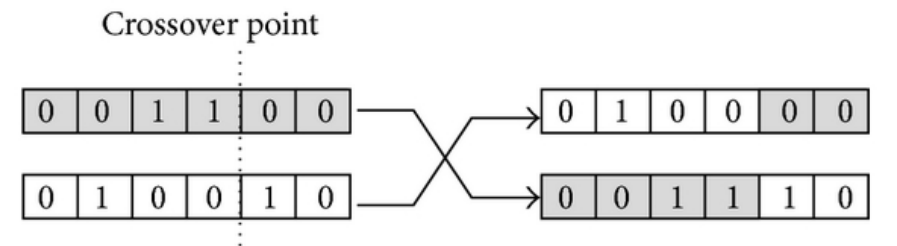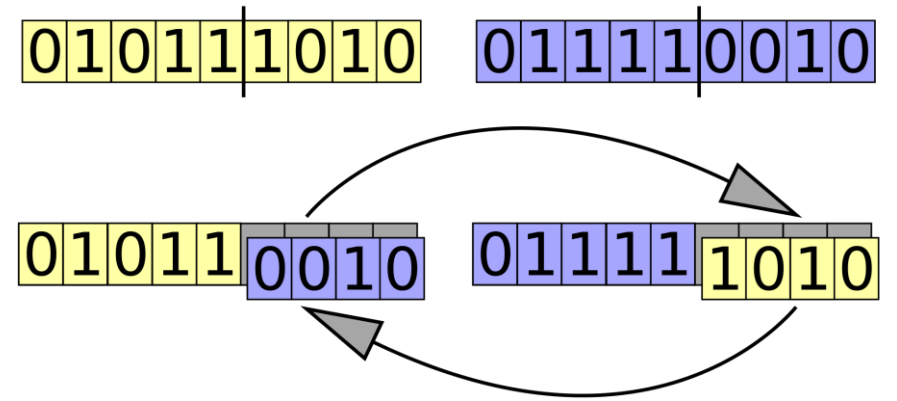
Two-point crossover
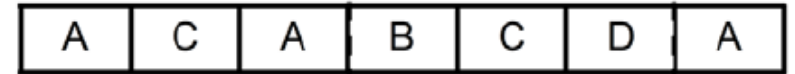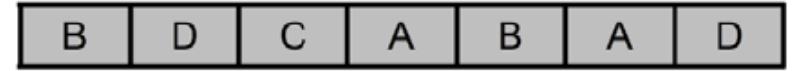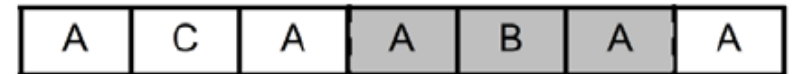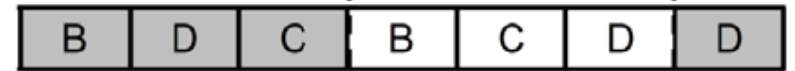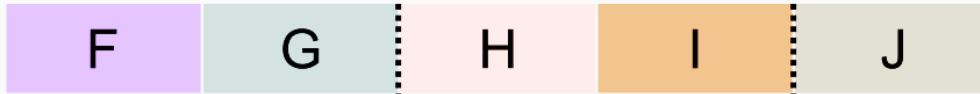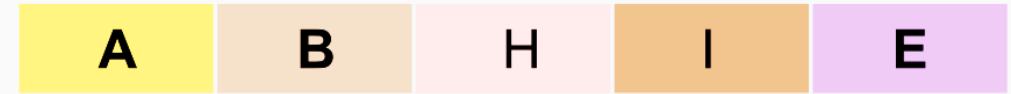
Multi Point Crossover

Example of Uniform Crossover

Consider two parent chromosomes:

- **Parent 1:** A B C D E

- **Parent 2:** 1 2 3 4 5

Suppose a randomly generated binary mask is 1 0 1 0 1 . Based on this mask:

- **Offspring 1:** A 2 C 4 E

- **Offspring 2** (using the inverted mask): 1 B 3 D 5

Mutation

- Definition and role of mutation: Introduces genetic diversity, helping avoid local minima.

- Types of mutation:
    - **Bit-flip mutation:** In binary chromosomes, flips bits (0 to 1 or 1 to 0).

- Mutation rate and its effect on the algorithm's performance.
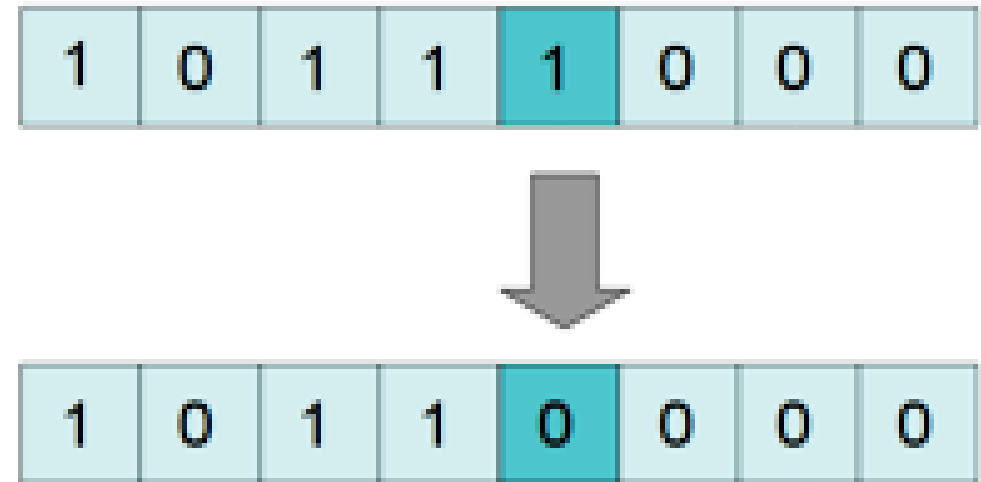
1	0	1	1	1	0	0	0

1	0	1	1	0	0	0	0

Operators Example

The Problem is to Maximize $f(x) = x^2$

Number	String	Fitness	% of the Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0

1.- *Roulette Wheel Selection*

$$P_s = \frac{f_i}{\sum_{j=1}^{d} f_j}$$

01000 10011 31% 01101

5% 14%
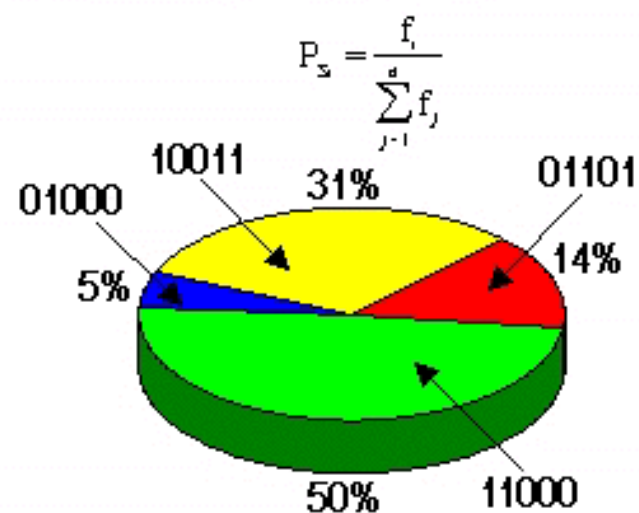
50% 11000

2.- *One-Point Crossover (SPX)*

$P_c \in [0.6 .. 1.0]$

parents	offspring
01|101(169)	01000(64)
11|000(576)	11101(841)

3.- *Mutation*

$P_m \in [0.001 .. 0.1]$

Mutate the first allele

0	1	0	0	0
4	3	2	1	0

1	1	0	0	0
4	3	2	1	0

fitness=64 ⇨ *fitness=576*
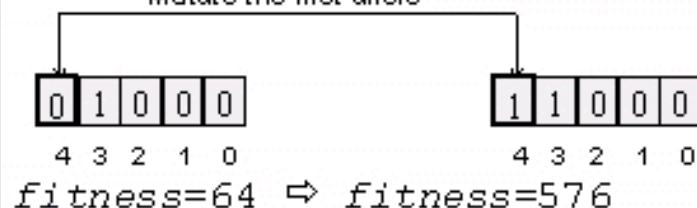
Fitness Evaluation

- **Role of fitness function: Measures the quality of a solution.**

- Designing a fitness function:
  - Must be problem-specific.
  - Should differentiate between good and poor solutions.

- Examples:
  - **Optimization problems:** Minimize or maximize a cost function.
  - **Scheduling:** Minimize delays or conflicts.

- Challenges: Designing a fitness function that guides the search effectively.

Fitness = f(Sequence)

Fitness

Residue 2

Residue 1
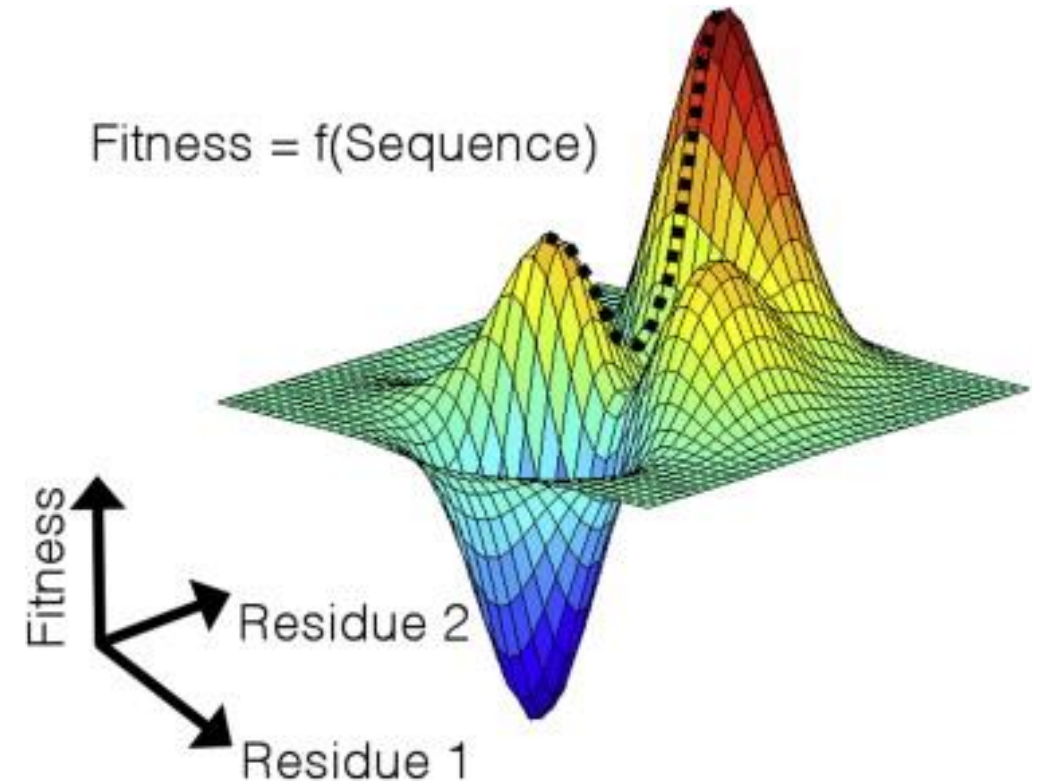
Stopping Conditions

- The evolutionary operators are iteratively applied in an EA until a stopping condition is satisfied.
  - Terminate when no improvement is observed over several consecutive generations.
  - Terminate when there is no change in the population.
  - Terminate when an acceptable solution has been found.
  - Terminate when the objective function slope is approximately zero.

Convergence Curves for GAs
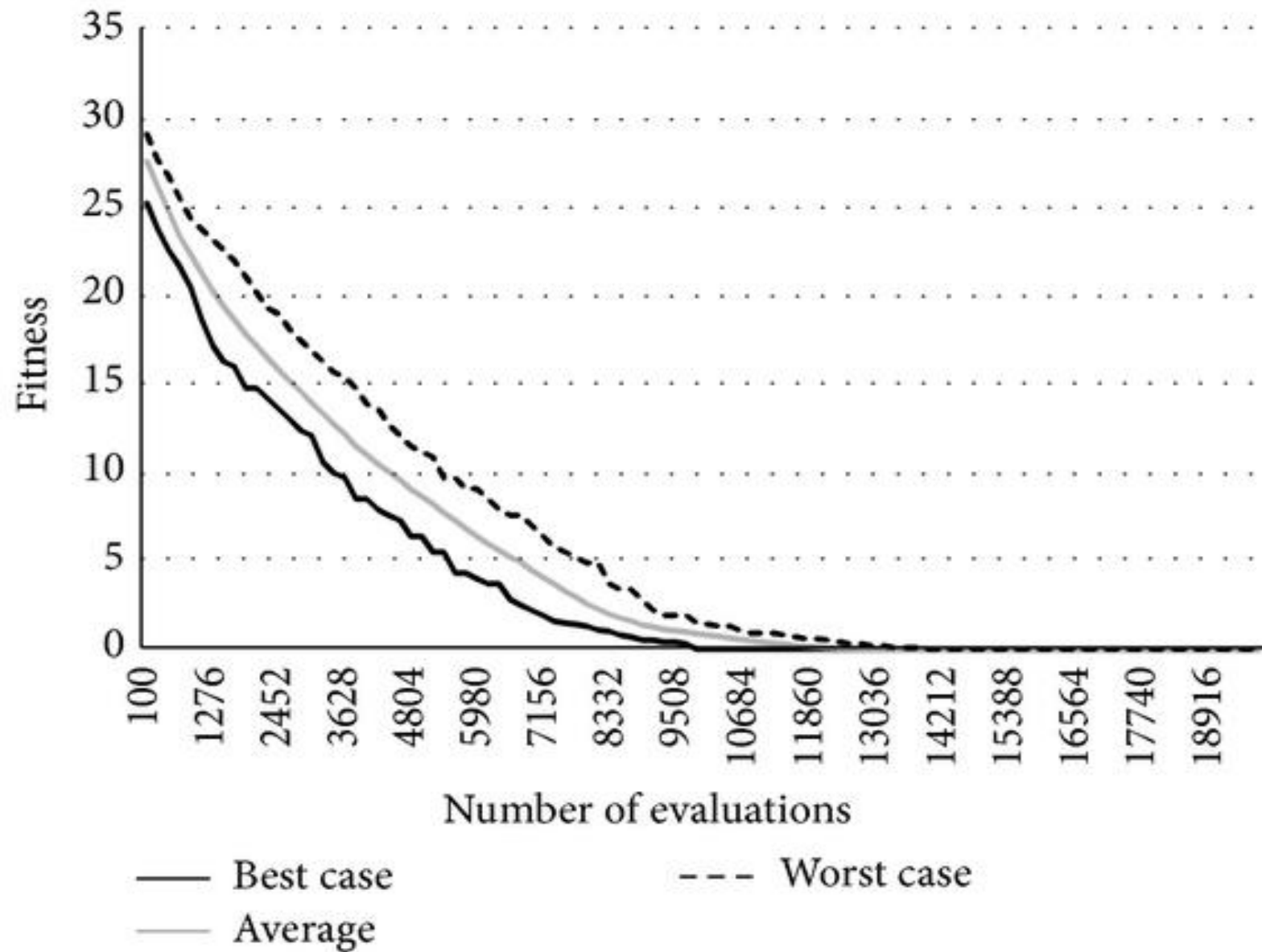
Advantages and Limitations of GAs

- Advantages:
    - Flexibility across various problem types.
    - Good for large search spaces.
    - Can avoid local optima in complex landscapes.

- Limitations:
    - Can be computationally intensive.
    - Risk of premature convergence.
    - Sensitive to parameter tuning (mutation rate, population size, etc.).

Applications of Genetic Algorithms

- **Optimization:** Resource allocation, parameter tuning.
- **Machine Learning:** Feature selection, neural network training.
- **Scheduling:** Job shop scheduling, course timetabling.
- **Game Development:** Evolving strategies, character behavior.
- **Biological Research:** Simulating genetic processes and protein folding.

Tools and Libraries for GAs

- Popular tools and libraries for implementing GAs:
  - **Python:** PyGAD
  - **MATLAB:** Global Optimization Toolbox

Summary
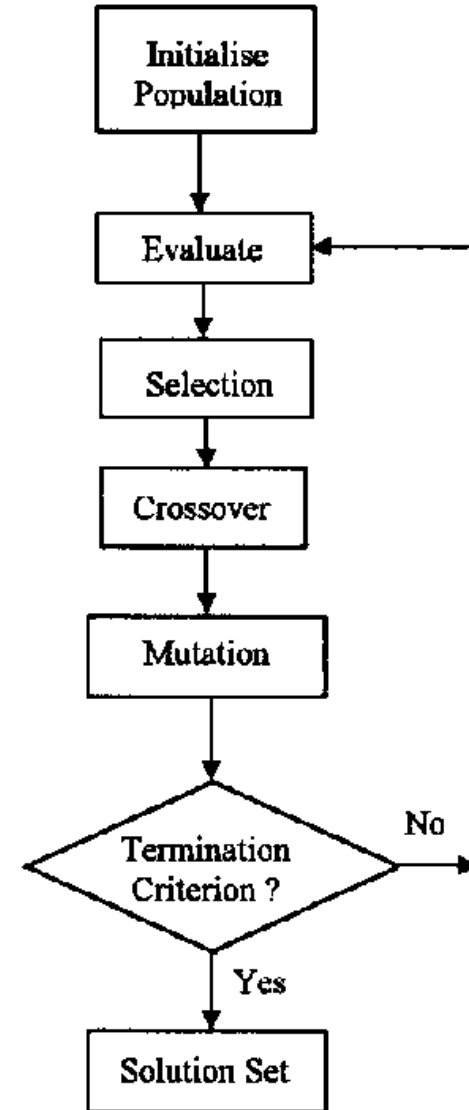
Example: Maximizing a Function of One Variable

- Consider the problem of maximizing the function. This example adapts the method of an example presented in Goldberg's book.

Graph of $f(x) = \frac{-x^2}{10} + 3x$

Example (cont.)

Graph of $f(x) = \frac{-x^2}{10} + 3x$

- For this example, we will encode x as a binary integer of length 5. Thus, the chromosomes for our genetic algorithm will be sequences of 0's and 1's with a length of 5 bits and have a range from 0 (00000) to 31 (11111).

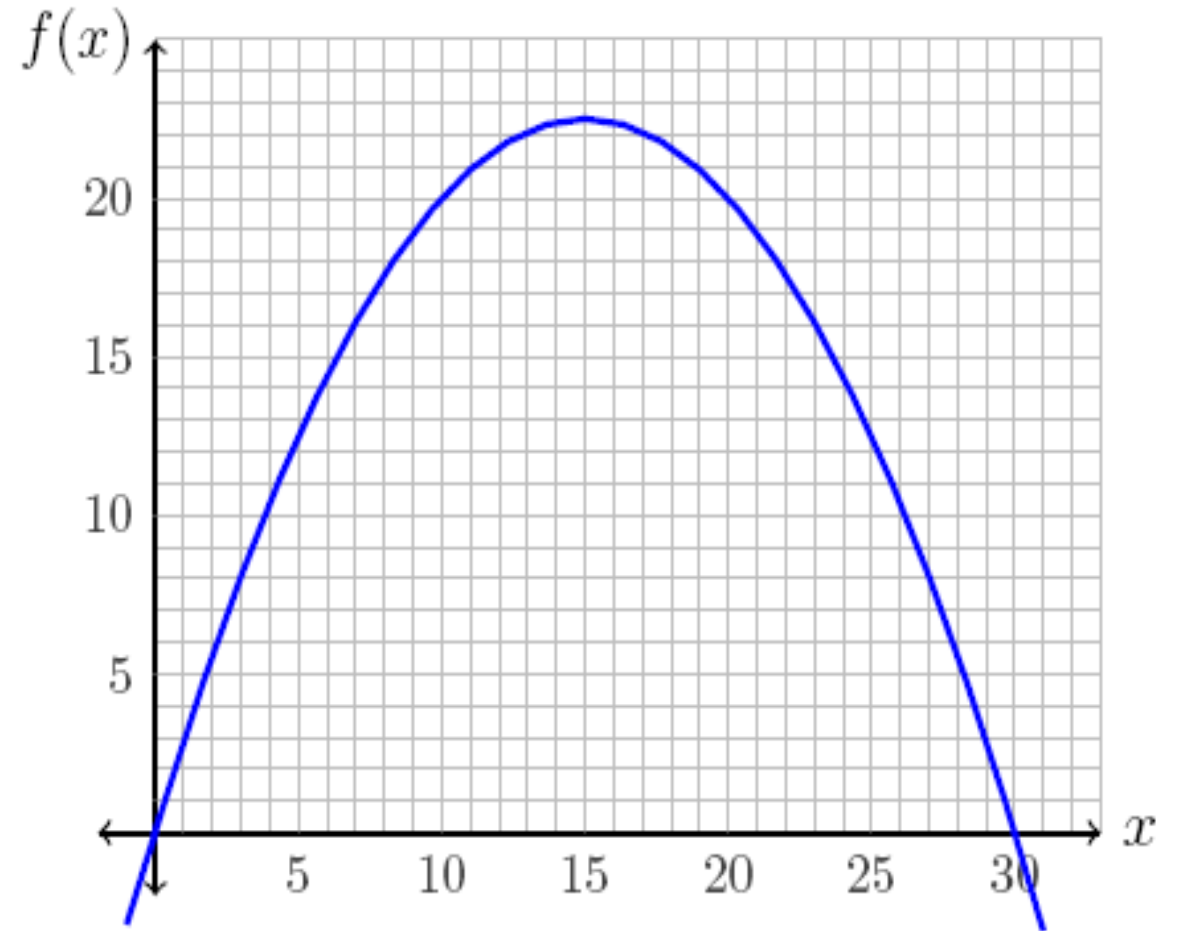- To begin the algorithm, we select an initial population of 10 chromosomes at random. We can achieve this by tossing a fair coin 5 times for each chromosome, letting heads signify 1 and tails signify 0.

- The resulting initial population of chromosomes is shown in Table

Initial Population

Chromosome Number	Initial Population	x Value	Fitness Value $f(x)$	Selection Probability
1	0 1 0 1 1	11	20.9	0.1416
2	1 1 0 1 0	26	10.4	0.0705
3	0 0 0 1 0	2	5.6	0.0379
4	0 1 1 1 0	14	22.4	0.1518
5	0 1 1 0 0	12	21.6	0.1463
6	1 1 1 1 0	30	0	0
7	1 0 1 1 0	22	17.6	0.1192
8	0 1 0 0 1	9	18.9	0.1280
9	0 0 0 1 1	3	8.1	0.0549
10	1 0 0 0 1	17	22.1	0.1497
	Sum		147.6	
	Average		14.76	
	Max		22.4	

Selection for mating

- We select the chromosomes that will reproduce based on their fitness values using the following probability:

$$P(\text{chromosome } i \text{ reproduces}) = \frac{f(x_i)}{\sum_{k=1}^{10} f(x_k)}$$
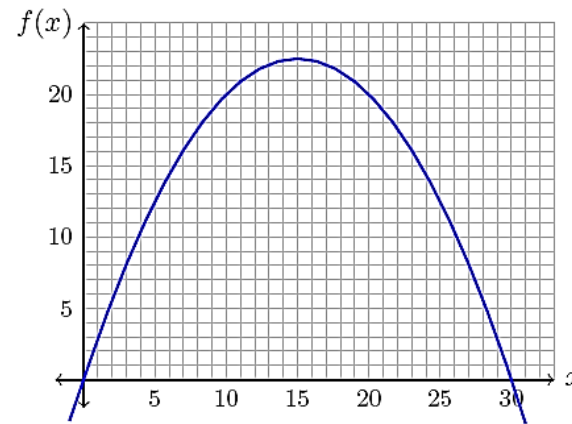
Reproduction and second generation

- Since our population has 10 chromosomes and each `mating' produces two offspring, we need 5 matings to create a new generation of 10 chromosomes. The selected chromosomes are displayed in Table.

- Lastly, each bit of the new chromosomes mutates with a low probability.

- For this example, we let the probability of mutation be 0.001. With 50 total transferred bit positions, we expect 500*0.001 = 0:05 bits to mutate.

Reproduction & Second Generation

Chromosome Number	Mating Pairs	New Population	x Value	Fitness Value $f(x)$
5	0 1|1 0 0	0 1 0 1 0	10	20
2	1 1|0 1 0	1 1 1 0 0	28	5.6
4	0 1 1 1|0	0 1 1 1 1	15	22.5
8	0 1 0 0|1	0 1 0 0 0	8	17.6
9	0 0 0 1|1	0 1 0 1 0	10	20
2	1 1 0 1|0	1 1 0 1 1	27	8.1
7	1 0 1 1 0	1 0 1 1 0	22	17.6
4	0 1 1 1 0	0 1 1 1 0	14	22.4
10	1 0 0|0 1	1 0 0 0 1	17	22.1
8	0 1 0|0 1	0 1 0 0 1	9	18.9
			Sum	174.8
			Average	17.48
			Max	22.5

Comparison

Graph of $f(x) = \frac{-x^2}{10} + 3x$

Initial Population

Chromosome Number	Initial Population	x Value	Fitness Value $f(x)$	Selection Probability
1	0 1 0 1 1	11	20.9	0.1416
2	1 1 0 1 0	26	10.4	0.0705
3	0 0 0 1 0	2	5.6	0.0379
4	0 1 1 1 0	14	22.4	0.1518
5	0 1 1 0 0	12	21.6	0.1463
6	1 1 1 1 0	30	0	0
7	1 0 1 1 0	22	17.6	0.1192
8	0 1 0 0 1	9	18.9	0.1280
9	0 0 0 1 1	3	8.1	0.0549
10	1 0 0 0 1	17	22.1	0.1497
	Sum		147.6	
	Average		14.76	
	Max		22.4	

Reproduction & Second Generation

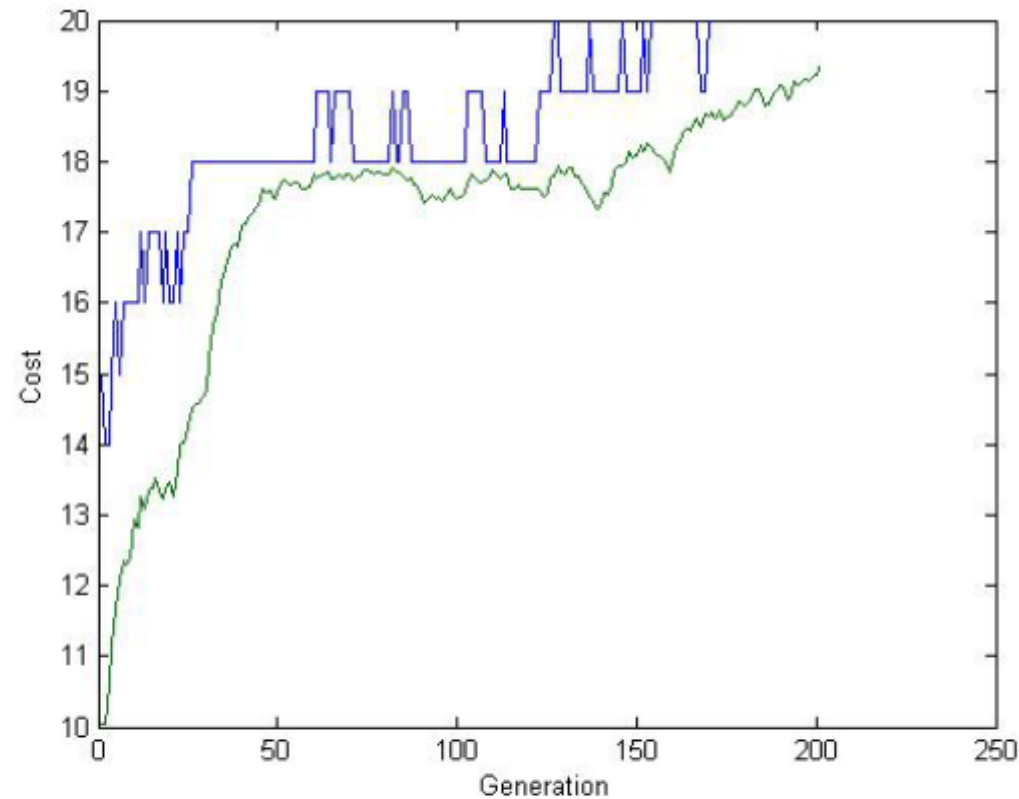| Chromosome Number	Mating Pairs	New Population	x Value	Fitness Value $f(x)$
5	0 1|1 0 0	0 1 0 1 0	10	20
2	1 1|0 1 0	1 1 1 0 0	28	5.6
4	0 1 1 1|0	0 1 1 1 1	15	22.5
8	0 1 0 0|1	0 1 0 0 0	8	17.6
9	0 0 0 1|1	0 1 0 1 0	10	20
2	1 1 0 1|0	1 1 0 1 1	27	8.1
7	1 0 1 1 0	1 0 1 1 0	22	17.6
4	0 1 1 1 0	0 1 1 1 0	14	22.4
10	1 0 0|0 1	1 0 0 0 1	17	22.1
8	0 1 0|0 1	0 1 0 0 1	9	18.9
			Sum	174.8
			Average	17.48
			Max	22.5

Run 1

Figure 2: The first run of a genetic algorithm maximizing the number of 1's in string of 20 bits. The blue curve is highest fitness, and the green curve is average fitness. Best solution: [11111111111111111111].
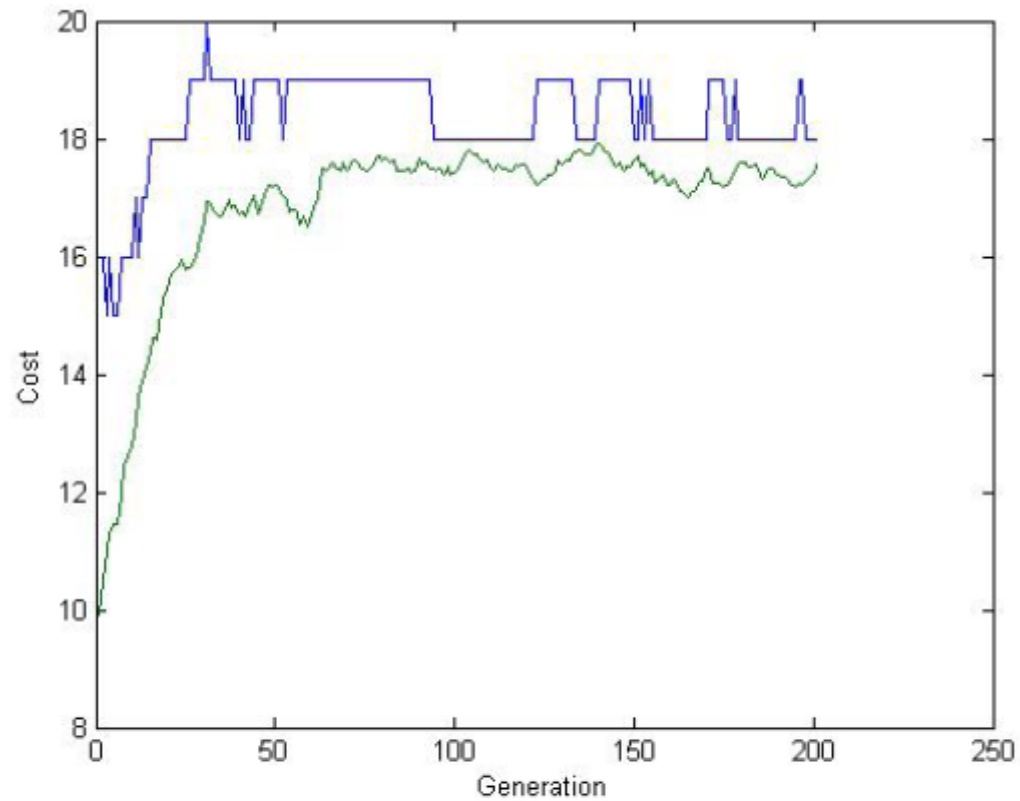
Run 2

Figure 3: The second run of a genetic algorithm maximizing the number of 1's in string of 20 bits. The blue curve is highest fitness, and the green curve is average fitness. Best solution: [11101111110111111111].
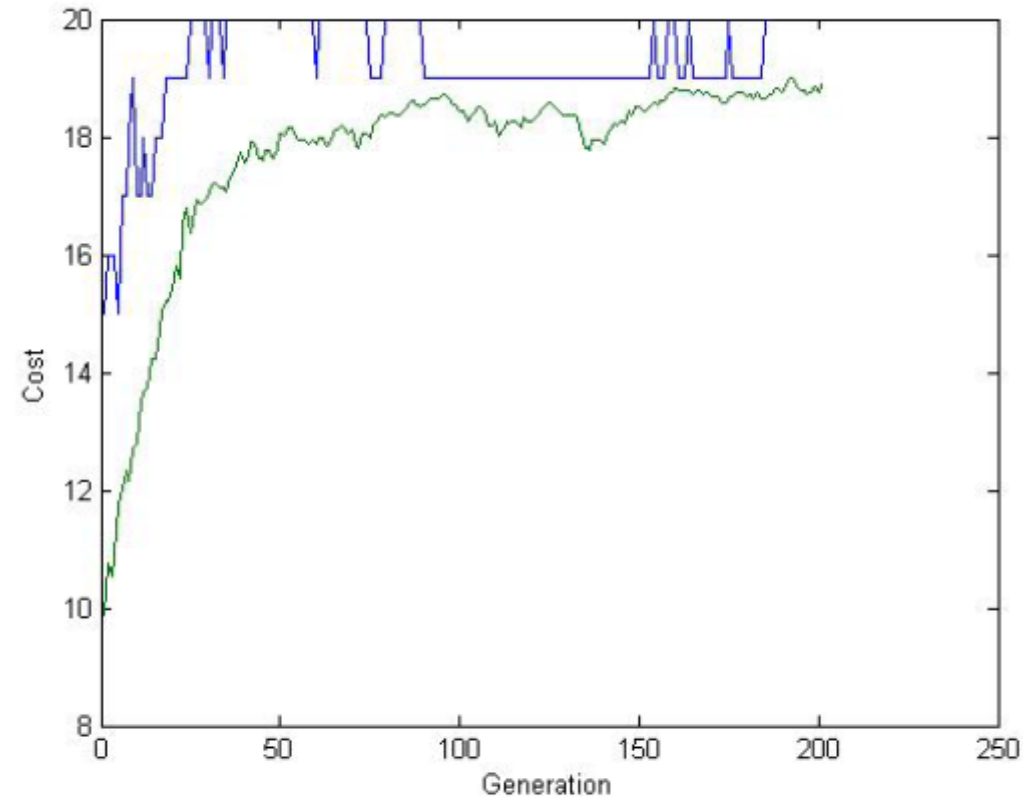
Run 3

Figure 4: The third run of a genetic algorithm maximizing the number of 1's in string of 20 bits. The blue curve is highest fitness, and the green curve is average fitness. Best solution: [11111111111111111111].

thank you