Artificial Intelligence

Beyond Classical Search

Genetic algorithms

- A genetic algorithm is a variant of stochastic beam search in which successor states are generated by combining two parent states rather than by modifying a single state.
- A successor state is generated by combining two parent states
- Start with *k* randomly generated states (population)
- A state is represented as a string over a finite alphabet (often a string of 0s and 1s)
- Evaluation function (fitness function). Higher values for better states.
- Produce the next generation of states by selection, *crossover*, and *mutation* (*genetic operators*)

Biology Concepts

- Population
- Fitness
- Selection
- Crossover
- Mutation

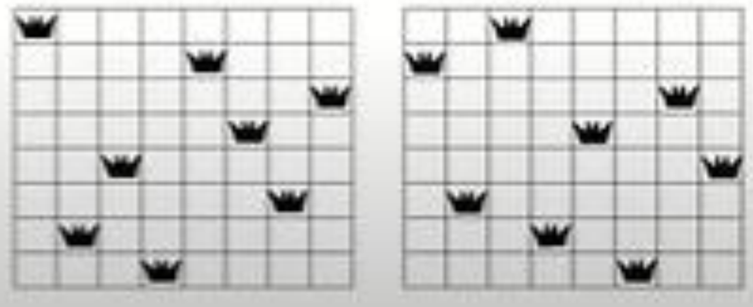
Biology Concepts: Population

Biology

- Collection of individuals.

Algorithm

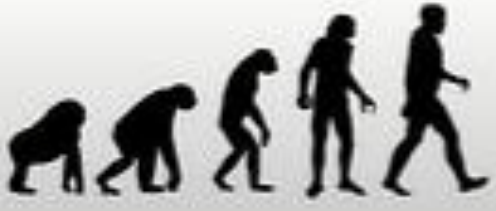
- Collection of states.



Biology Concepts: Fitness

Biology Concepts: Selection

Biology	Algorithm
Selecting species that are the most biologically fit.	Selecting states that are closest to the solution (Fittest).



Biology Concepts: Crossover

Biology

generation

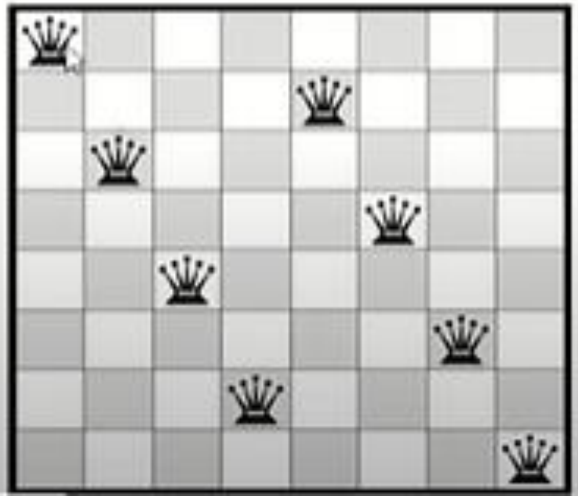
Algorithm

Interchanging values between selected states

Biology Concepts: Mutation

8- Queens Problem

- Arrange 8 queens on a standard chess board in such a way that no queen attacks each other.



Solving 8-Queens using Genetic Algorithm

- Step 1: Representing individuals.
- Step 2: Generating an initial Population.
- Step 3: Applying a Fitness Function.
- Step 4: Selecting parents for mating in accordance to their fitness.
- Step 5: Crossover of parents to produce new generation.
- Step 6: Mutation of new generation to bring diversity.
- Step 7: Repeat until solution is reached.

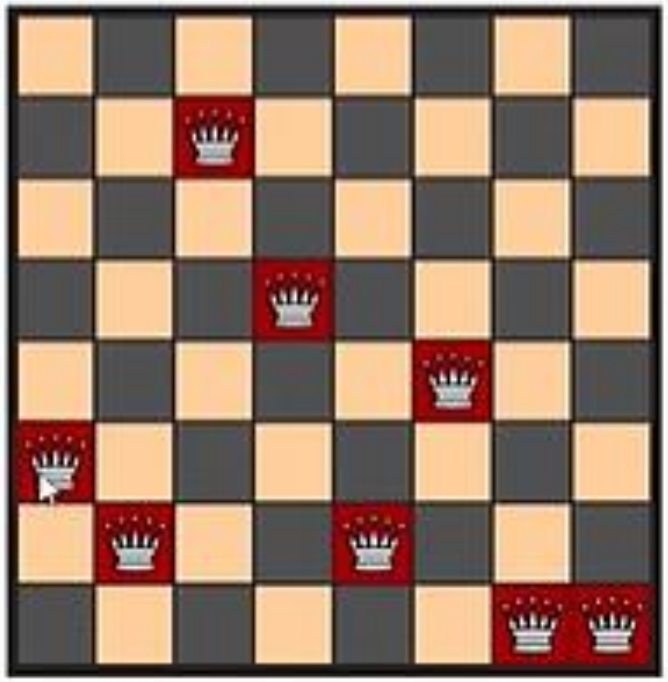
Step 1: Representing Individuals/States

- Formulate an appropriate method to represent individuals of a population.
- Array.
- Index: Column.
- Value: Row.

1-8

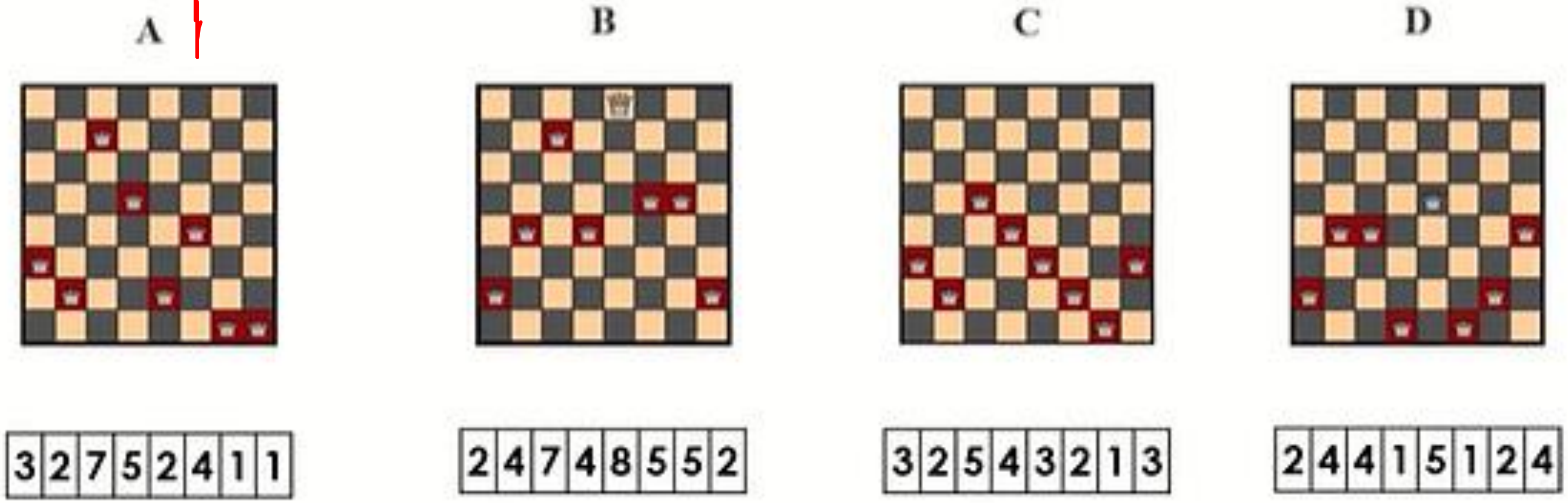
3	2	7	5	2	4	1	1

9



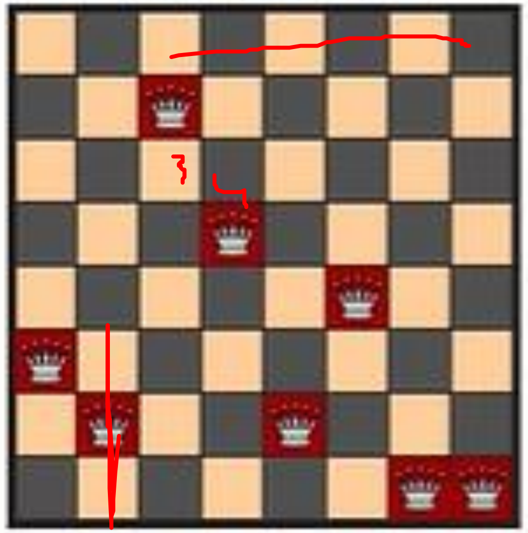
Step 2: Generate Initial Population

- Generate random arrangements of 8 queens on a standard chess board.



Step 3: Apply Fitness Function

Individual



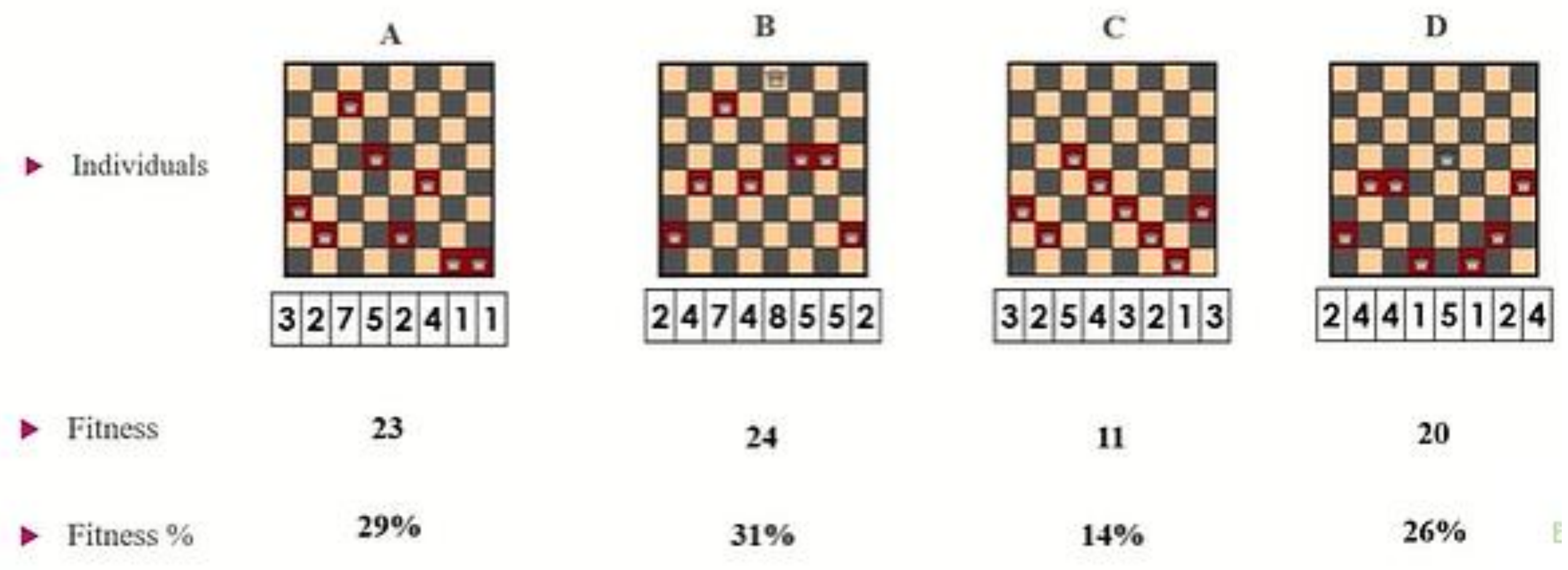
3	2	7	5	2	4	1	1

Fitness = No. of non attacking pairs

- Queen 1: 6
- Queen 2: 5
- Queen 3: 4
- Queen 4: 3
- Queen 5: 3
- Queen 6: 2
- Queen 7: 0
- Queen 8: 0

Total 23

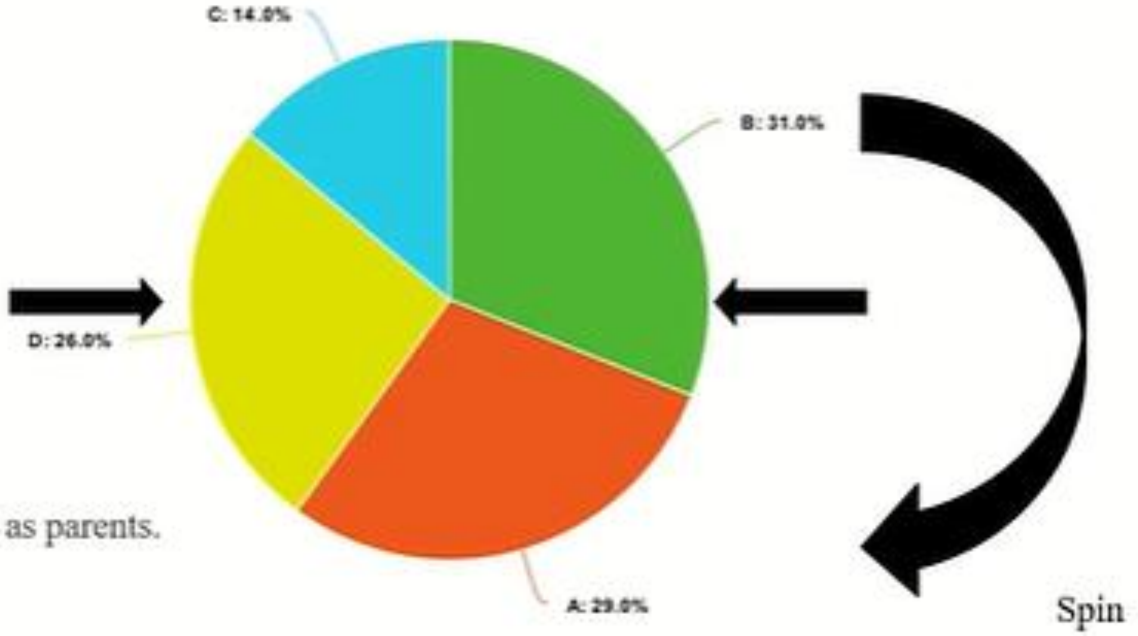
Step 3: Apply Fitness Function (contd.)

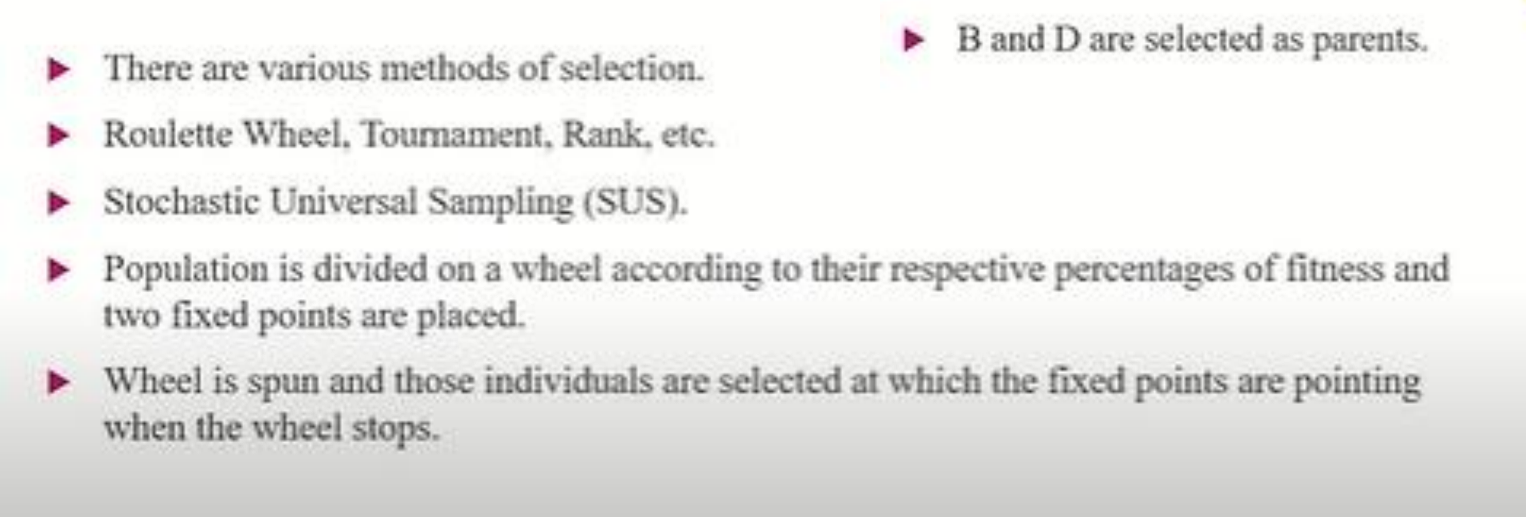


Total:78

Step 4: Selection

Proportionate selection or roulette wheel selection gives individuals **with higher fitness values** a higher chance of being selected as parents, mimicking the concept of "survival of the fittest" in the natural evolutionary process.

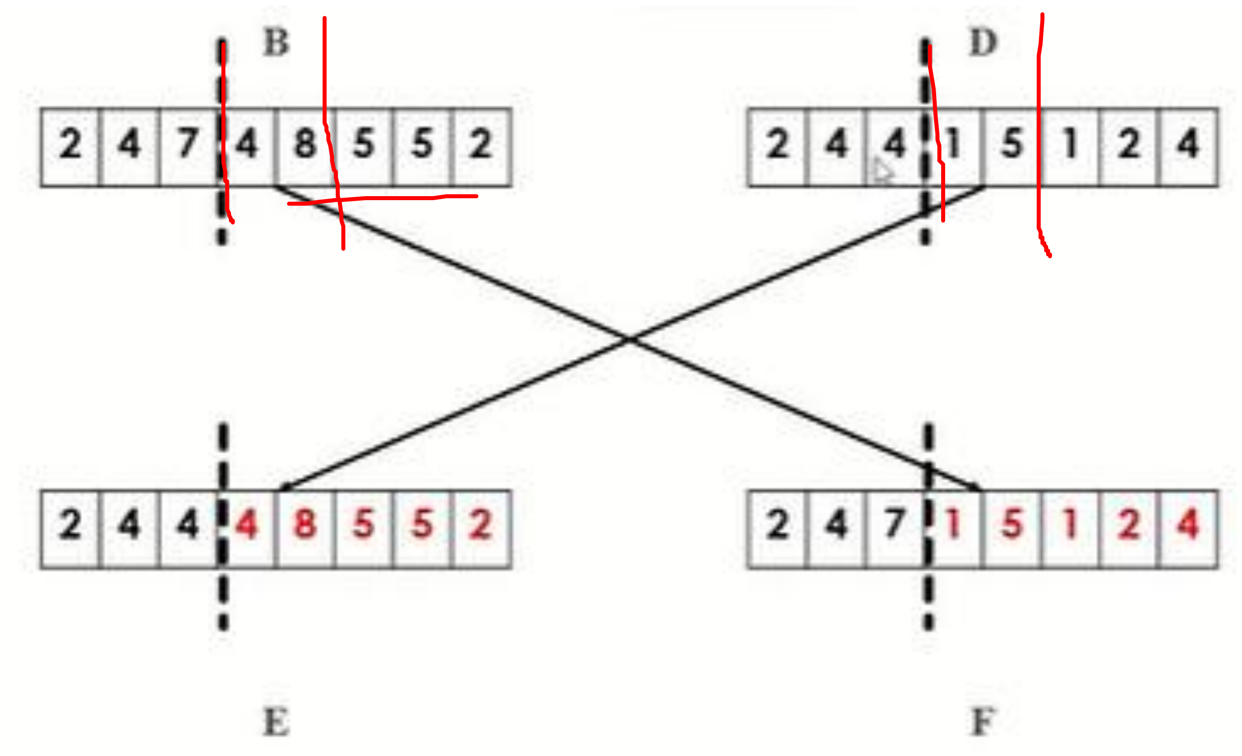




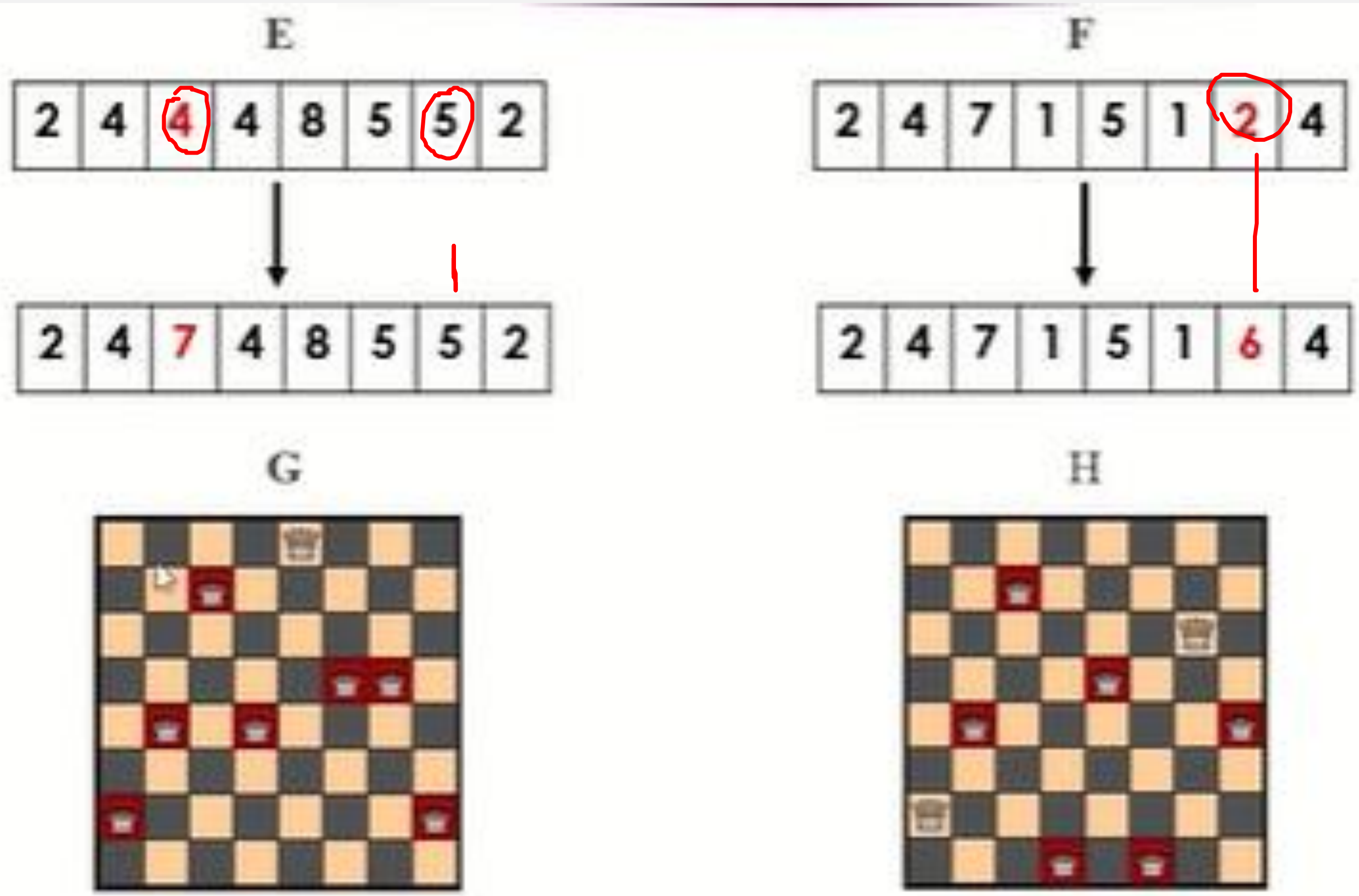
- B and D are selected as parents.
- There are various methods of selection.
- Roulette Wheel, Tournament, Rank, etc.
- Stochastic Universal Sampling (SUS).
- Population is divided on a wheel according to their respective percentages of fitness and two fixed points are placed.
- Wheel is spun and those individuals are selected at which the fixed points are pointing when the wheel stops.

The key difference between SUS and traditional roulette wheel selection is that **SUS selects multiple parents in a single pass, rather than selecting one parent at a time.**

Step 5: Crossover



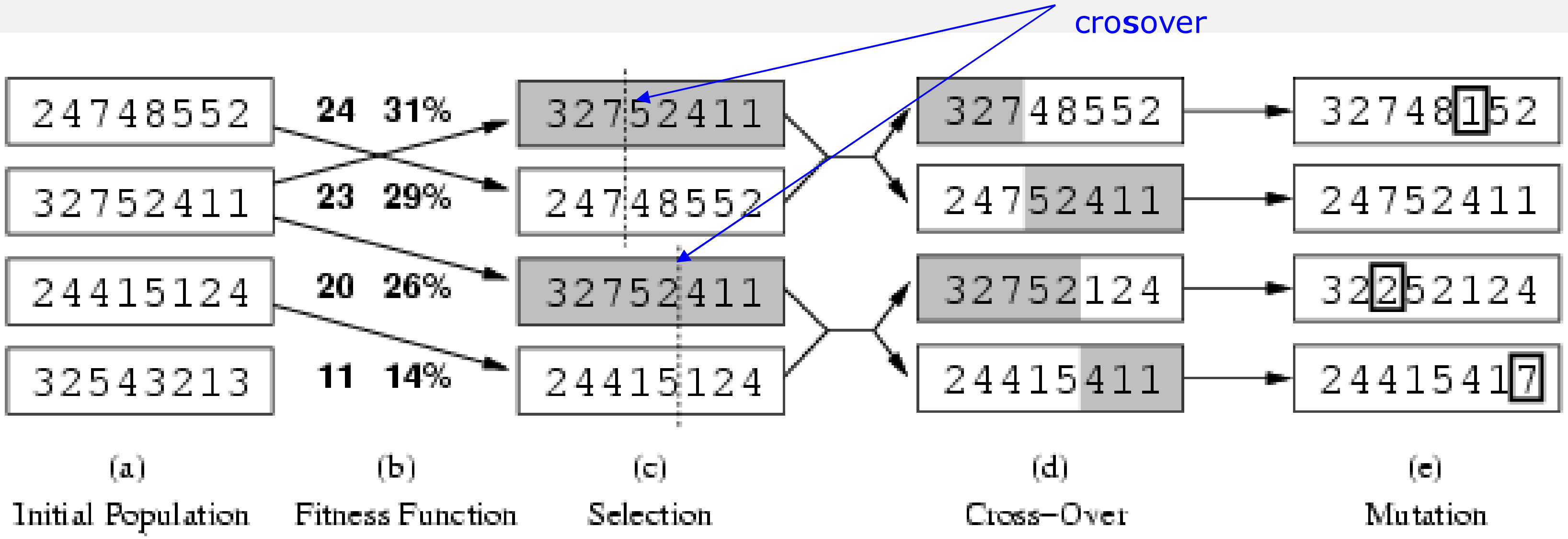
Step 6: Mutation



Step 7: Repeat

- All steps are repeated until best solution is reached.
- Best solution = Highest fitness score (28 in this case).

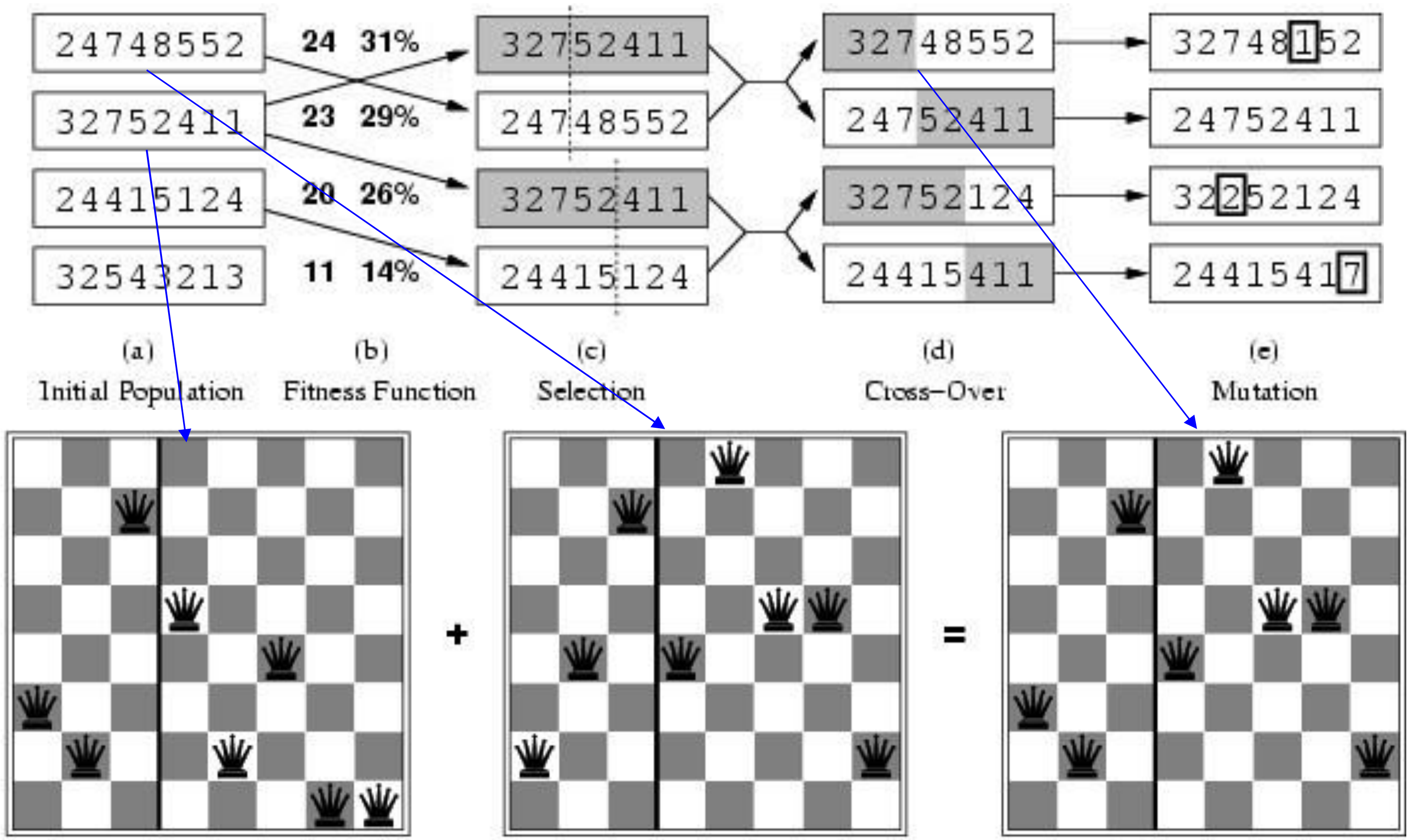
Genetic algorithms



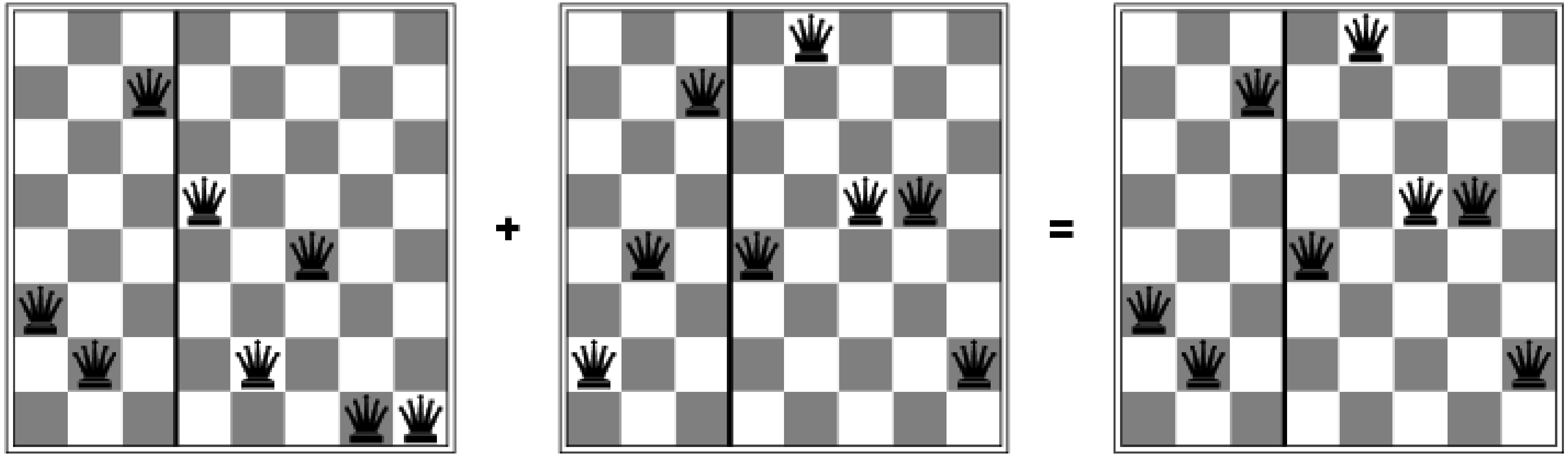
Example 8-Queen

- Fitness function: number of non-attacking pairs of queens (min = 0, max = 28)
- 24/(24+23+20+11) = 31%
- 23/(24+23+20+11) = 29% etc

8-Queen Example



Genetic algorithms



Algorithm

```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
          FITNESS-FN, a function that measures the fitness of an individual

  repeat
     new_population ← empty set
     for i = 1 to SIZE(population) do
        x ← RANDOM-SELECTION(population, FITNESS-FN)
        y ← RANDOM-SELECTION(population, FITNESS-FN)
        child ← REPRODUCE(x, y)
        if (small random probability) then child ← MUTATE(child)
        add child to new_population
     population ← new_population
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to FITNESS-FN
```

```
function REPRODUCE(x, y) returns an individual
  inputs: x, y, parent individuals

  n ← LENGTH(x); c ← random number from 1 to n
  return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c + 1, n))
```

Figure 4.8 A genetic algorithm. The algorithm is the same as the one diagrammed in Figure 4.6, with one variation: in this more popular version, each mating of two parents produces only one offspring, not two.

Applications: Genetic algorithms

- DNA Analysis
- Image Processing
- Vehicle Routing
- Neural Network

❑ **Solving Difficult Problems**

GAs prove to be an efficient tool to provide **usable near-optimal solutions** in a short amount of time.

❑ **Failure of Gradient Based Methods**

In most real-world situations we have a very complex problem as it suffer from an inherent tendency of getting stuck at the local optima as shown here.

❑ **Getting a Good Solution Fast**

Some difficult problems like the Travelling Salesperson Problem (TSP), have real-world applications like path finding and VLSI Design.

Solving 8-Queens using Genetic Algorithm

- Step 1: Representing individuals.
- Step 2: Generating an initial Population.
- Step 3: Applying a Fitness Function.
- Step 4: Selecting parents for mating in accordance to their fitness.
- Step 5: Crossover of parents to produce new generation.
- Step 6: Mutation of new generation to bring diversity.
- Step 7: Repeat until solution is reached.

Example: Knapsack Problem (Capacity=12kg)

Item	Weight (kg)	Value
A	5	10
B	3	5
C	7	9
D	2	8

Binary representation of chromosomes: genotype
Integer representation of chromosomes : phenotype

Problem is that which item should be kept in the knapsack so as it will maximize knapsack value without breaking knapsack

Knapsack Problem

*0 represents absence of item
*1 represent the presence of item
State Space=2^4

Population

A	B	C	D
0	1	1	0
0	0	1	1
0	1	1	1
1	0	1	0

Fitness

Fitness Function for 1:
Weight=3+ 7=10
Value=5+9=14

Fitness Function for 2:
Weight=7+ 2=9
Value=8+9=17

Fitness Function for 3:
Weight=3+ 7+2=12
Value=5+9+8=22

Fitness Function for 4:
Weight=5+ 7=12
Value=10+9=19

Selection

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

*0 represents absence of item
*1 represent the presence of item
State Space=2^4

Knapsack Problem

Population

	A	B	C	D
C1	0	1	1	0
C2	0	0	1	1
C3	0	1	1	1
C4	1	0	1	0

Fitness

Fitness Function for 1:
Weight=3+ 7=10
Value=5+9=14

Fitness Function for 2:
Weight=7+ 2=9
Value=8+9=17

Fitness Function for 3:
Weight=3+ 7+2=12
Value=5+9+8=22

Fitness Function for 4:
Weight=5+ 7=12
Value=10+9=19

Selection

Roulette Wheel selection: the state that has the highest fitness value gets larger share of the wheel

0	1	1	1	C3
0	0	1	1	C2
1	0	1	0	C4
0	1	1	1	C3

Knapsack Problem

*0 represents absence of item
*1 represent the presence of item
State Space=2^4

Crossover: Randomly select the position around which gene would be exchanged.

C3	0	1	1	1
C2	0	0	1	1
C4	1	0	1	0
C3	0	1	1	1

Selection

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

Cross-Over

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

Mutation

Knapsack Problem

*0 represents absence of item
*1 represent the presence of item
State Space=2^4

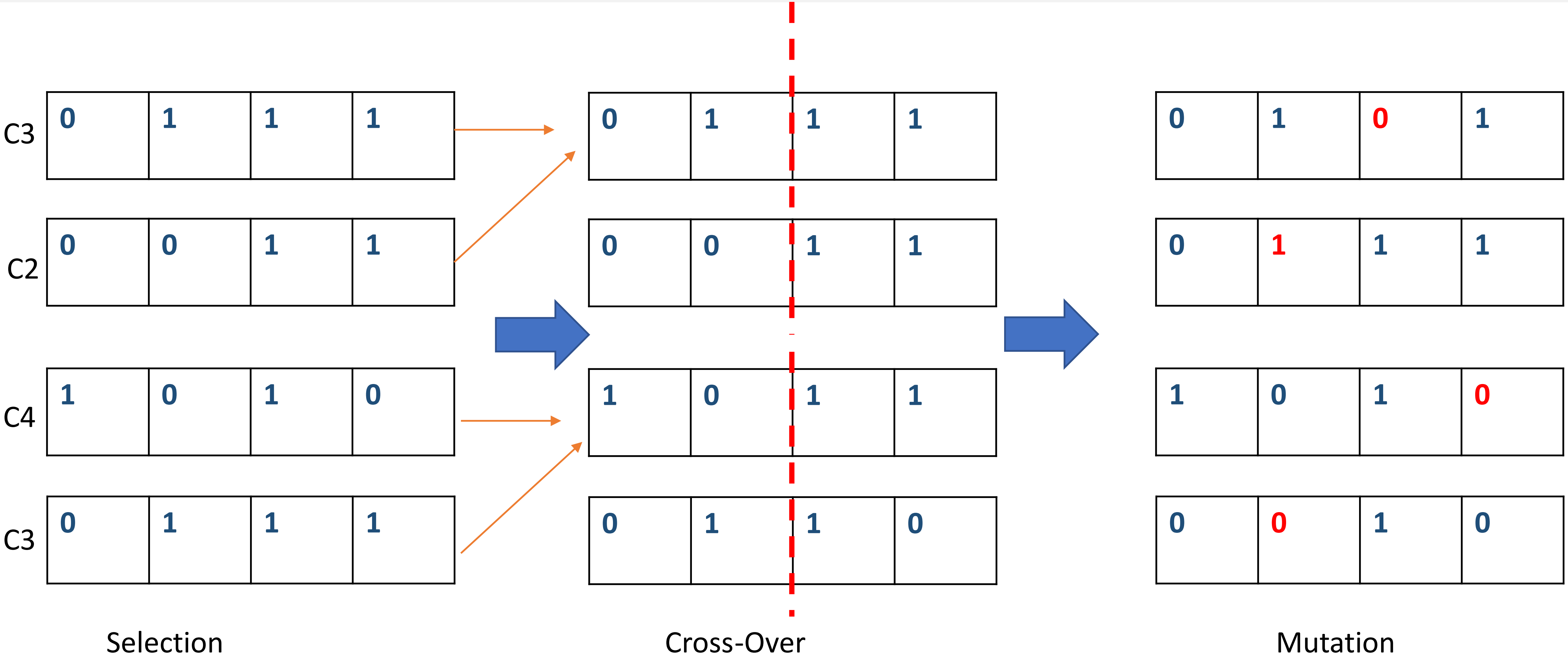
Introduces the diversity within the population so that search algorithm does not necessarily get stuck at local maxima.

20

Mutation

Knapsack Problem

*0 represents absence of item
*1 represent the presence of item
State Space=2^4



Genetic algorithms

- Genetic algorithms combine an uphill tendency with random exploration and exchange of information among parallel search threads
- Advantages come from “crossover”, which raise the level of granularity
- Does not require any derivative information which may not be available for many real-world problems
- Gives n no of Solution not just a single solution
- Useful when search space is very large

Linear Rank Example

QUESTION 2: Suppose there are 10 chromosomes with finesses as shown in table. What will be the selection probability according to the proportionate and linear rank selection methods?

Chromosome No.	Fitness fi	Proportionate $p_i = f_i / \sum_{j=1}^{n} f_j$	Linear Rank	Sort	$nf_i = (P - r_i) + 1$	r	Linear Rank Calculations $np_i = nf_i / \sum_{j=1}^{n} nf_j$
A	50	50/1000=0.05	0.055	250	(10-1)+1=10	1	10/55= 0.182
B	25	0.025	0.036	140	9	2	0.164
C	25	0.025	0.018	125	8	3	0.145
D	100	0.1	0.109	110	7	4	0.127
E	75	0.075	0.073	100	6	5	0.109
F	125	0.125	0.145	100	5	6	0.091
G	250	0.250	0.182	75	4	7	0.073
H	110	0.110	0.127	50	(10-8)+1=3	8	(3/55)= 0.055
I	140	0.140	0.164	25	2	9	0.036
J	100	0.1	0.091	25	1	10	0.018
Total	1000	1	1	1000	55		1

2 4 5

6

2

3

Qs 3. In genetic algorithm, a generation has 50 chromosomes, out of which 6 fittest ones are placed in the next generation without any competition from the offspring. The rest of the chromosomes of the next generation are chosen from the pool of parents and offspring. Suppose 12 chromosomes of the offspring make it to the new generation. What is the generation gap and elitism value. Duplicates are not allowed.

- Attendance will be marked based on the answer to this question.
- Write the answer in the comments on GCR.