AI Course

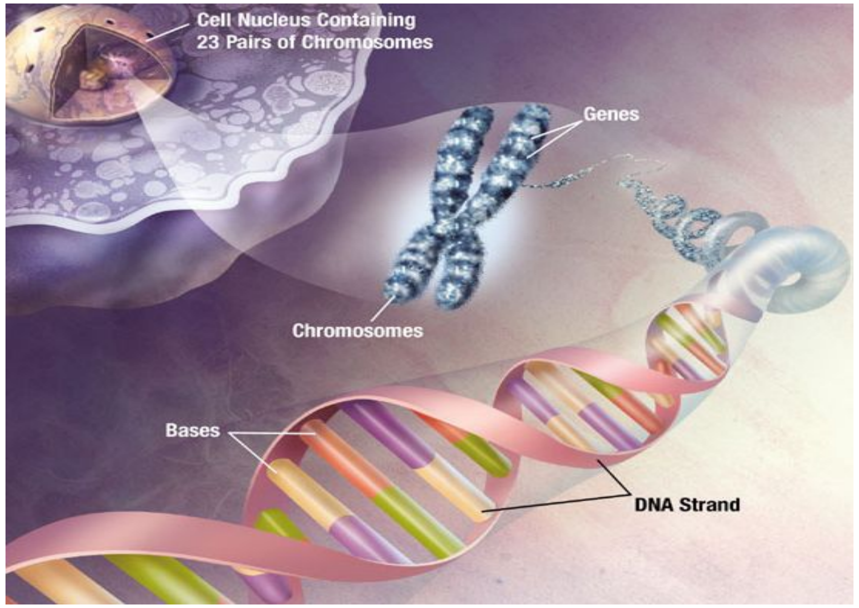
Dr. Mürsel Taşgın

Genetic Algorithm

Genetic Algorithm - *Introduction*

- Genetic algorithm (GA) is a search heuristic algorithm, inspired by nature and Darwin's theory of natural evolution

- <u>A brief history:</u>
  - 1957 – Alex Fraser (geneticist) – Simulation of artificial selection of organisms
  - 1970 – Fraser & Burnell, 1973-Crosby (biologist) – Computer simulation of evolution
  - 1975 – John Holland – (computer scientist) – Adaptation in Natural and Artificial Systems



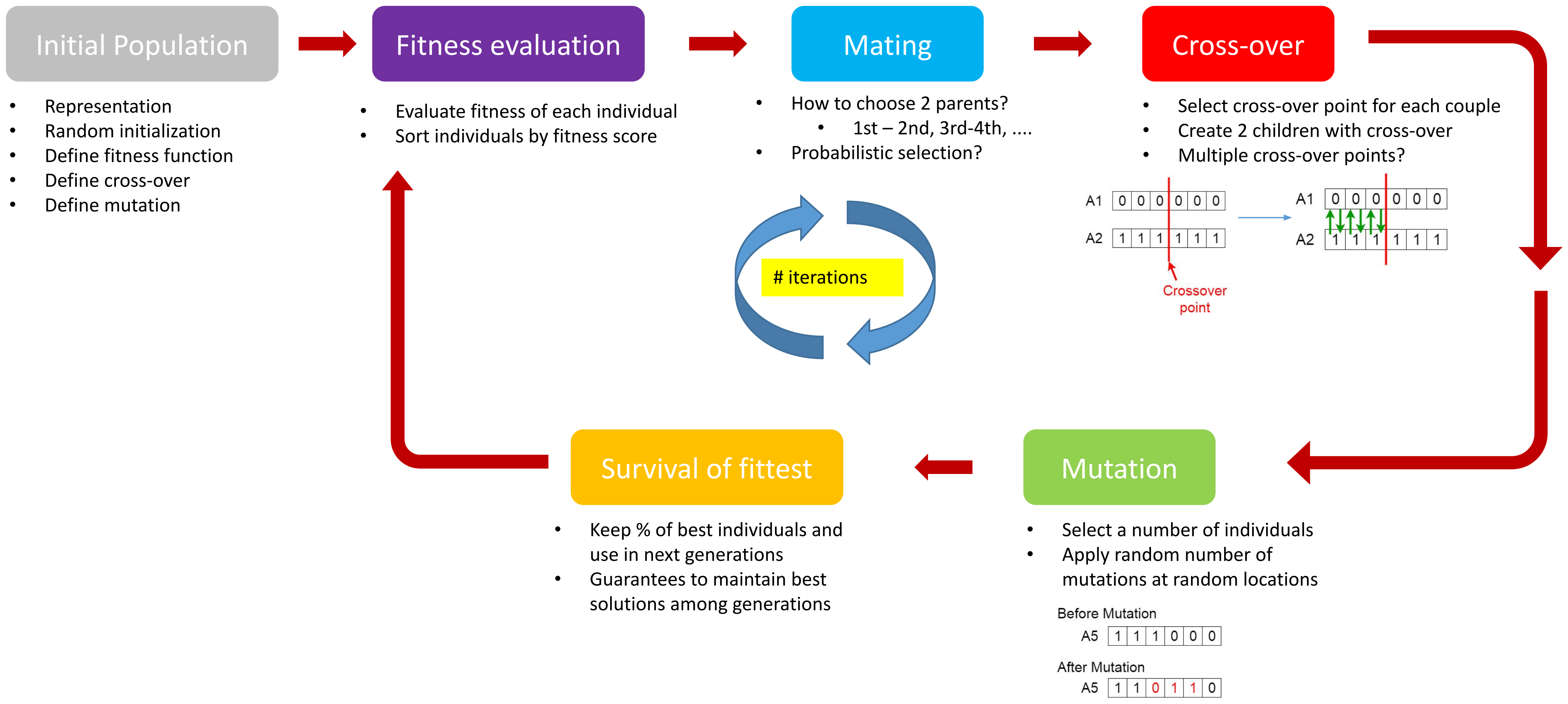
Genetic Algorithm - *Introduction*

- In GA, proposed solutions are represented by individuals, individuals form a population or generation
- Representation of the solution is encoded in the chromosomes of each individual
- Genetic operations of cross-over, mutation, survival of the fittest, etc. are used to generate better and better generations (i.e. Better solutions)
- At each iteration, individuals (solutions) are evaluated using a

Genetic Algorithm - *Questions*

- What do we aim with GA?
- Can it limit search space?
- Is it able to optimize solution?
- Does it converge (minimize error)?
- Is it brute-force?
- Any limitations?

Genetic Algorithm – *Overview of flow*



Initial Population

- Representation
- Random initialization
- Define fitness function
- Define cross-over
- Define mutation

Fitness evaluation

- Evaluate fitness of each individual
- Sort individuals by fitness score

Mating

- How to choose 2 parents?
  - 1st – 2nd, 3rd-4th, ...
- Probabilistic selection?

Cross-over

- Select cross-over point for each couple
- Create 2 children with cross-over
- Multiple cross-over points?

A1	0	0	0	0	0	0
A2	1	1	1	1	1	1

Mutation

- Select a number of individuals
- Apply random number of mutations at random locations

Before Mutation

A5	1	1	1	0	0	0

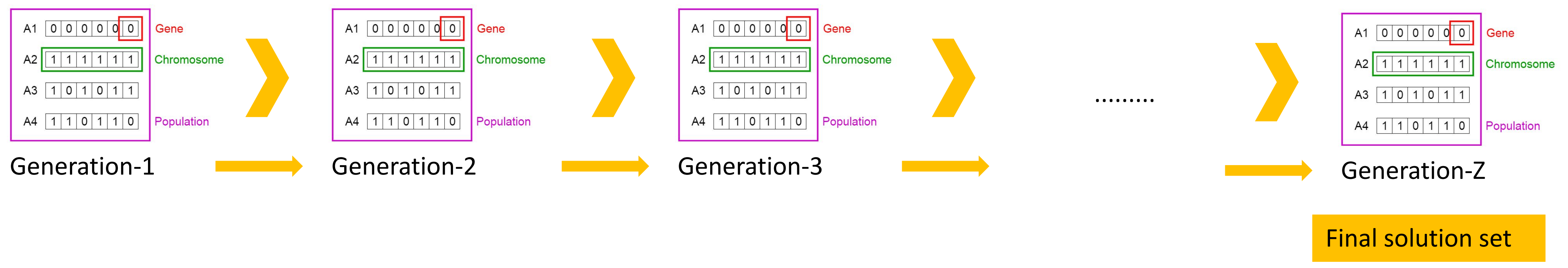
After Mutation

A5	1	1	0	1	1	0

Survival of fittest

- Keep % of best individuals and use in next generations
- Guarantees to maintain best solutions among generations

Genetic Algorithm – *Generations*

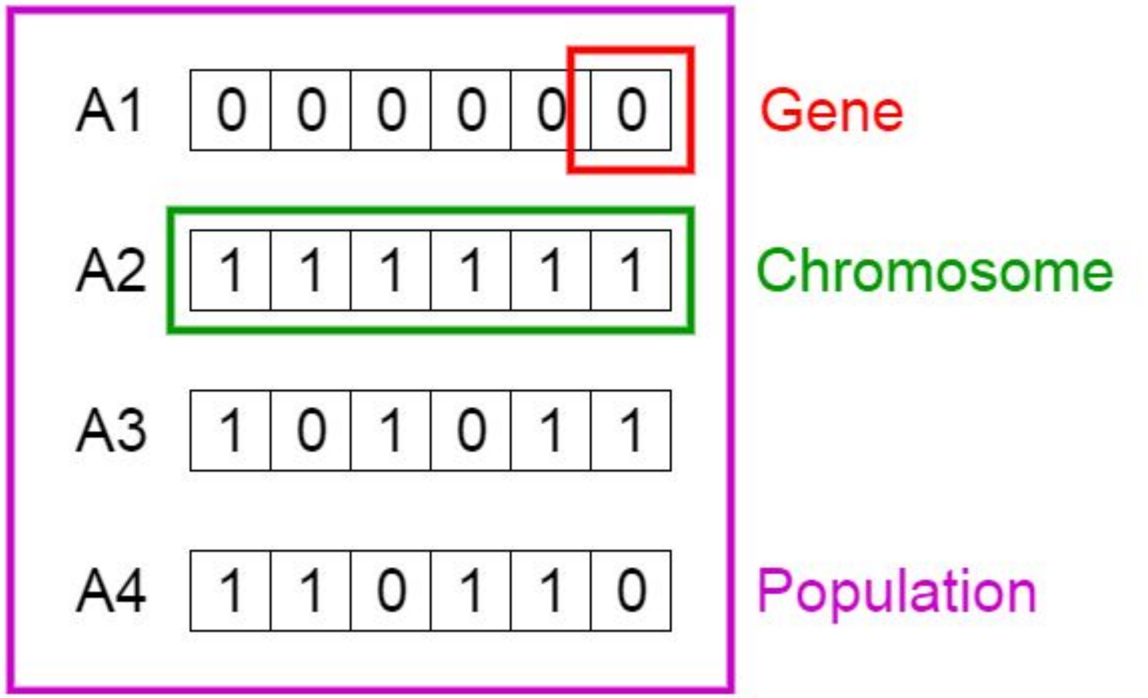


Generations:

- Like people populations, we start with initial population (Generation-1) and reproduce new individuals (children) for next generation
- Two parents in *Generation(t)* will have two children in *Generation(t+1)*
- Assumption: Each generation will have new individuals, no individual will live to next generation *(exception: survival of the fittest)*
- For reproduction, we use genetic operations like cross-over, mutation, survival of the fittest

- We need to evaluate (*score*) each individual according to our solution criteria (fitness function)

Genetic Algorithm – *Initial population*

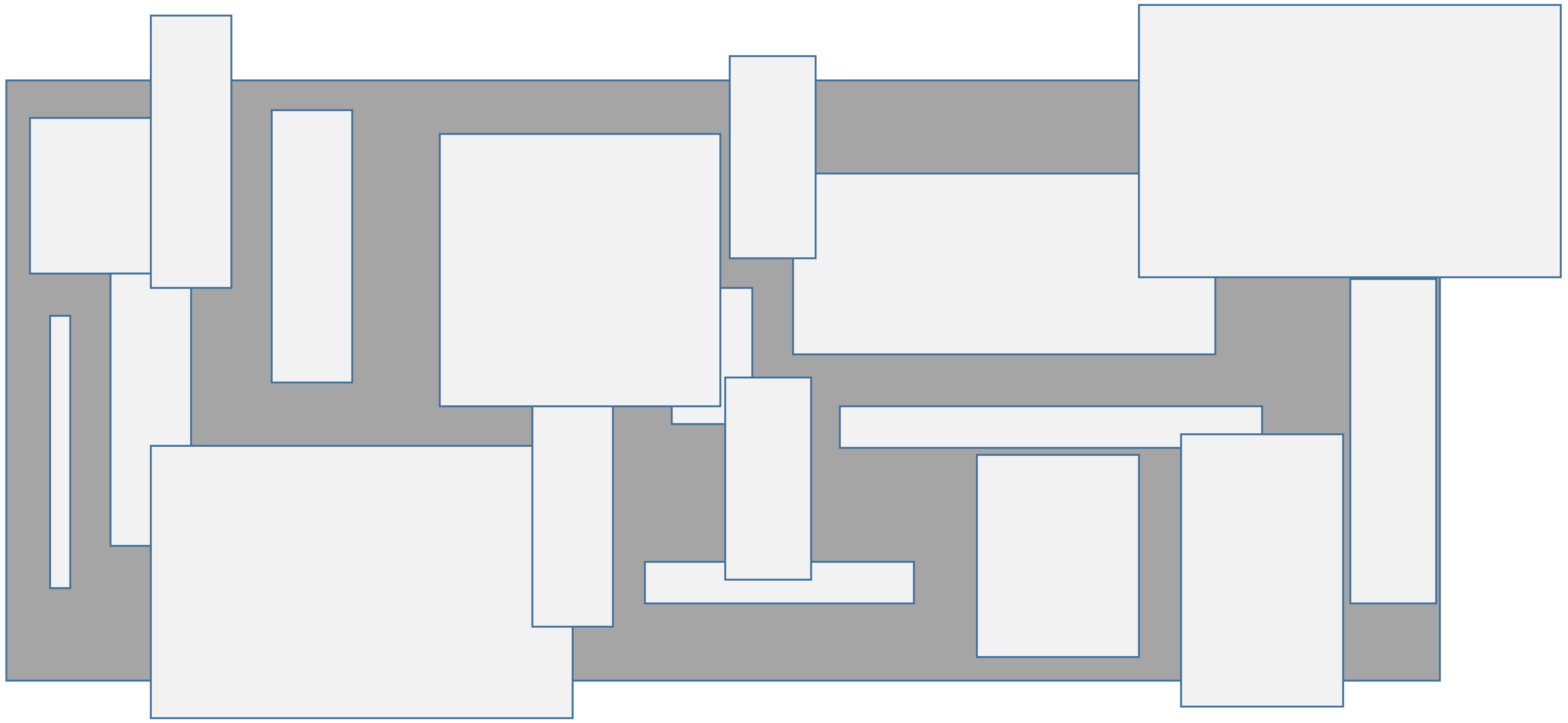


Representation:

- We need to find a suitable way to represent a candidate solution
- Solutions are encoded in chromosomes of individuals
- i.e., in a layout problem, chromosomes can represent X,Y coordinates of each object location
- We will have ***N*** number of individuals who form the population

We can use random initialization!

Genetic Algorithm – *Initial population*

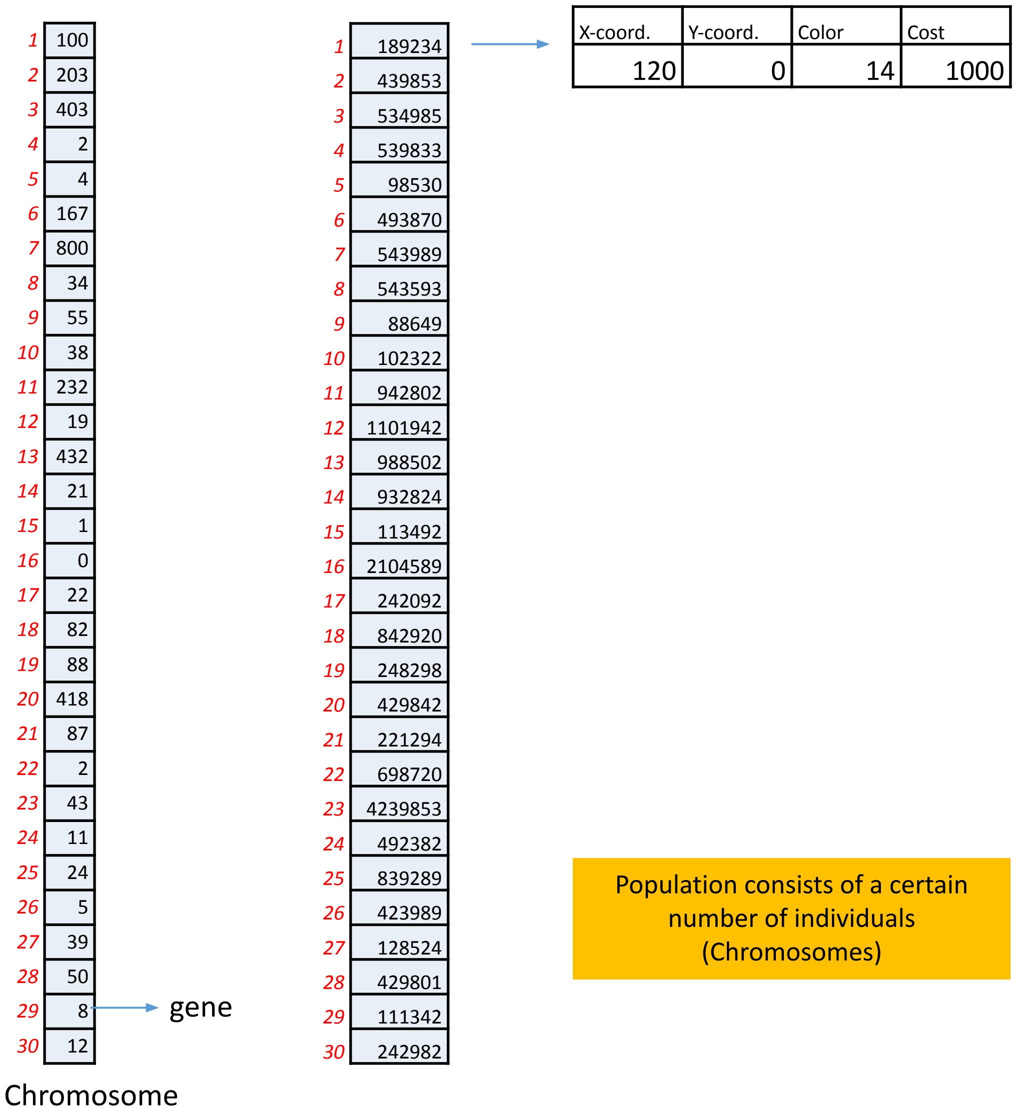


Rectangle Layout Problem – Initialization: *Randomly put on board*

Genetic Algorithm – *Initial population*

Representation:

- Representation is one the most critical parts of GA
- Each individual should hold a candidate solution of the problem
- Cross-over should not cause loss of information of the solution
- Representation should be suitable for genetic operations *(cross-over, mutation, etc.)*



1	100
2	203
3	403
4	2
5	4
6	167
7	800
8	34
9	55
10	38
11	232
12	19
13	432
14	21
15	1
16	0
17	22
18	82
19	88
20	418
21	87
22	2
23	43
24	11
25	24
26	5
27	39
28	50
29	8
30	12

1	189234
2	439853
3	534985
4	539833
5	98530
6	493870
7	543989
8	543593
9	88649
10	102322
11	942802
12	1101942
13	988502
14	932824
15	113492
16	2104589
17	242092
18	842920
19	248298
20	429842
21	221294
22	698720
23	4239853
24	492382
25	839289
26	423989
27	128524
28	429801
29	111342
30	242982

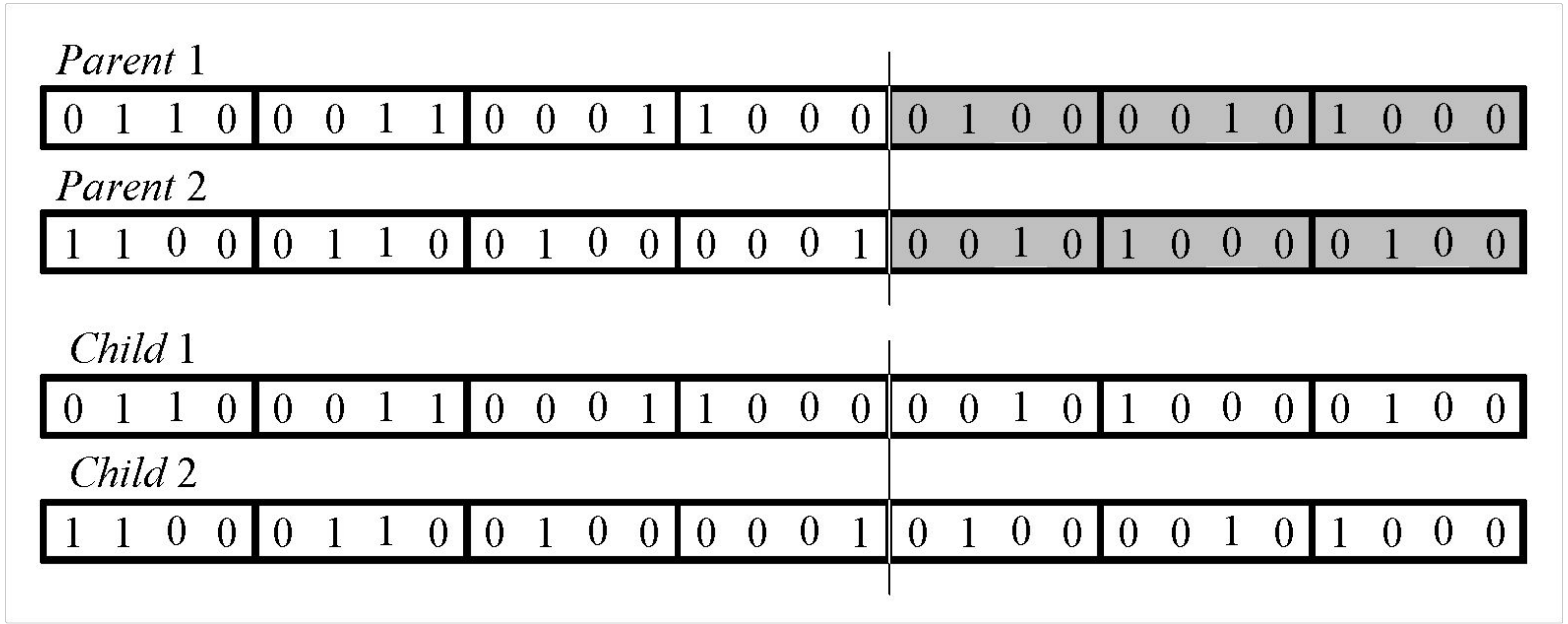
X-coord.	Y-coord.	Color	Cost
120	0	14	1000

Population consists of a certain number of individuals (Chromosomes)

Genetic Algorithm – *Initial population*

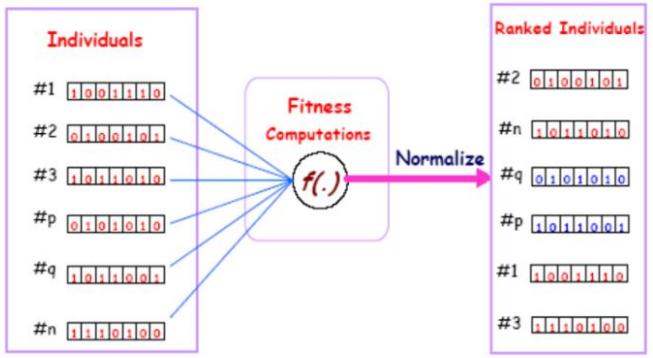
Chromosomes could be:

- Bit strings `(0101 ... 1100)`
- Real numbers `(43.2 -33.1 ... 0.0 89.2)`
- Permutations of element `(E11 E3 E7 ... E1 E15)`
- Lists of rules `(R1 R2 R3 ... R22 R23)`
- Program elements `(genetic programming)`
- ... any data structure ...



Genetic Algorithm – *Fitness Function*

- Fitness function evaluates the fitness of each individual solution candidate
- Fitness function will produce a single numeric score fitness score
- Fitness scores quantifies, how desirable, how good the individual is against the constraints or criteria of solution
- In biology, we can define a fitness function that can tell how healthy an individual is



Genetic Algorithm – *Fitness Function*

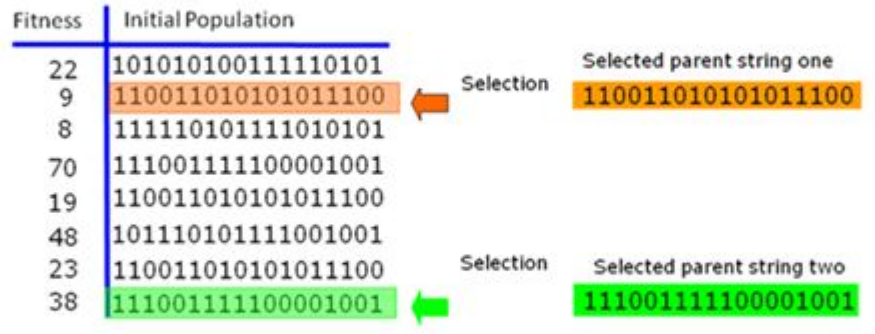
- We should write the fitness function to optimize our constraints
  - List the constraints
  - Try to formulize evaluation of constraints
  - Combine all subscores to a single numeric score
- Fitness functions are custom for each GA, it should be defined for our need
- Fitness scores should be calculated for each individual
- Individuals should be sorted according to fitness scores

1	203	100	100	100	3	5	5
2	100	203	203	203	203	203	203
3	403	403	403	403	403	403	403
4	2	2	4	2	2	2	2
5	4	4	2	4	4	4	4
6	167	167	167	167	167	167	800
7	800	800	800	800	800	800	167
8	34	34	34	34	34	34	34
9	55	55	55	55	55	38	55
10	38	38	38	38	38	55	38
11	232	232	232	418	232	232	232
12	19	19	21	19	19	19	19
13	432	432	432	432	432	432	432
14	21	21	19	21	21	21	21
15	1	1	1	1	22	1	1
16	0	0	0	0	0	0	0
17	22	22	22	22	22	22	22
18	82	82	82	82	88	82	82
19	88	88	88	88	82	88	88
20	418	418	418	232	418	418	418
21	87	87	87	87	87	87	87
22	2	2	2	2	2	2	2
23	43	43	43	43	43	43	43
24	11	11	11	11	11	11	11
25	5	24	24	24	24	24	24
26	24	5	5	5	5	5	5
27	39	39	39	39	39	39	39
28	50	50	50	50	50	50	50
29	8	8	8	8	8	8	8
30	12	12	12	12	12	12	12
Sort them all!	103	98	97	88	70	60	58

Genetic Algorithm – *Selection & Mating*

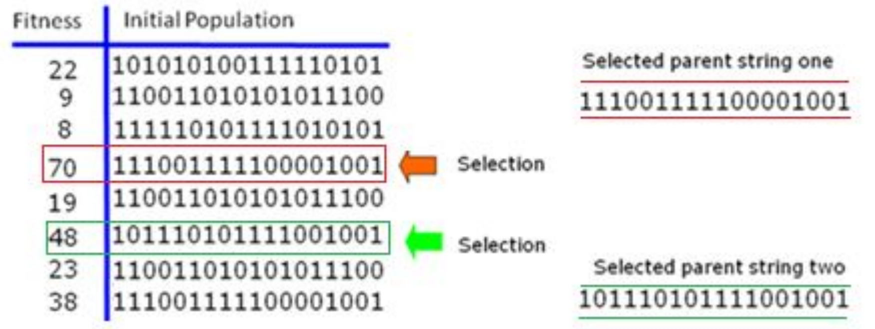
- How to choose 2 parents to create 2 children?
  - Random selection
  - Select according to fitness from top-to-bottom (1st – 2nd, 3rd – 4th, ...)
  - Select with probability of fitness function

$$p_s(\mathbf{m}_i) = \frac{F(\mathbf{m}_i)}{\sum_{j=1}^{n} F(\mathbf{m}_j)}$$



Fitness	Initial Population
22	101010100111110101
9	110011010101011100
8	111110101111010101
70	111001111100001001
19	110011010101011100
48	101110101111001001
23	110011010101011100
38	111001111100001001

Random selection

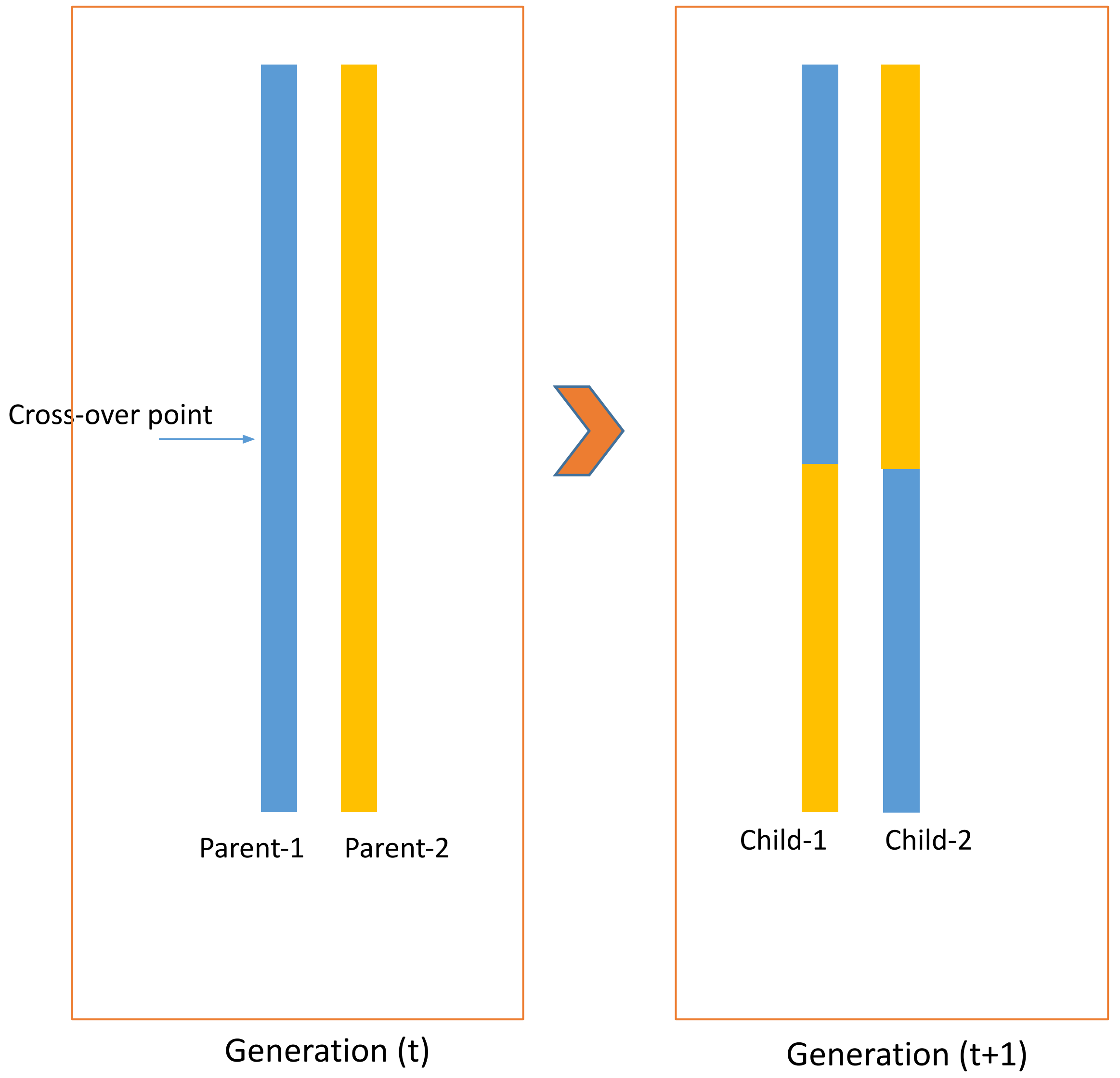


Fitness	Initial Population
22	101010100111110101
9	110011010101011100
8	111110101111010101
70	111001111100001001
19	110011010101011100
48	101110101111001001
23	110011010101011100
38	111001111100001001

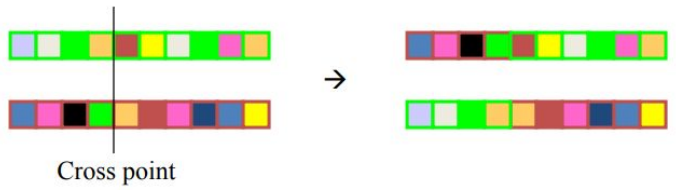
Selection by fitness score

Genetic Algorithm – *Cross-over*

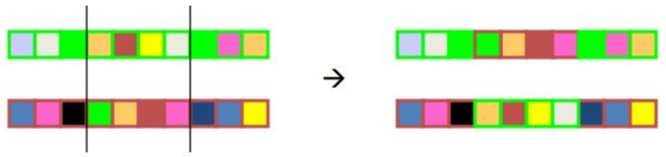
- Select one or more cross-over point
- Exchange genes between chromosomes (individuals)
- Create 2 new individuals for next generation
- All the chromosomes in the population must re-produce (create 2 new children)



- Single point crossover



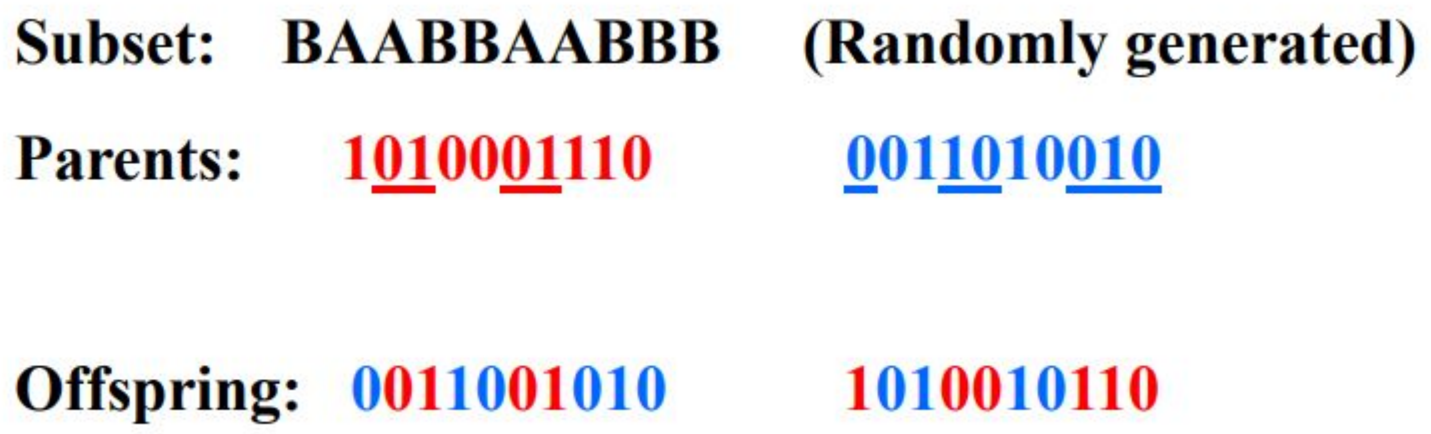
- Two point crossover (Multi point crossover)



Genetic Algorithm – *Cross-over*

- Many variants of cross-over possible

- Example: Uniform cross-over
  - A random subset is chosen
  - The subset is taken from parent 1 and the other bits from parent 2



Subset: BAABBAABBB (Randomly generated)

Parents: $1\underline{01}00\underline{01}110$ $\underline{0}01\underline{10}10\underline{010}$

Offspring: 0011001010 1010010110

Genetic Algorithm - *Mutation*

- Randomly change a bit of information in chromosomes
- Can create new information, new solutions!

Creativity?

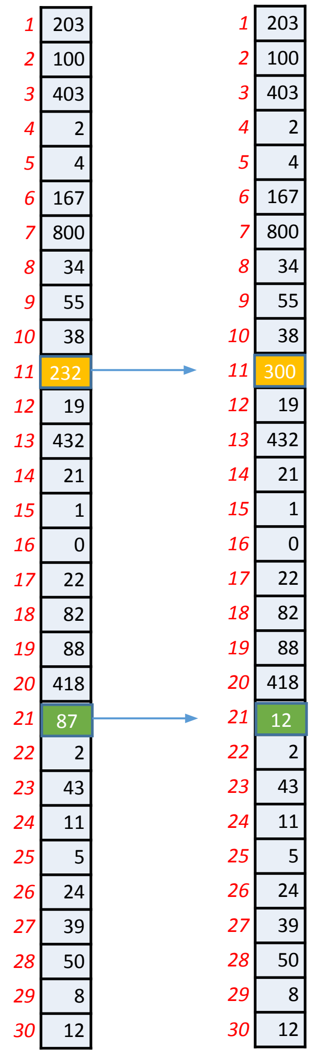
- It can be applied to only a group of chromosomes
  - Select a percentage of chromosomes randomly
  - Select a number of genes and change them randomly

- Harmful mutations may kill!
  - Like biological life, mutation may lead to unintended consequences Death!
- Mutation can also lead better solutions which did not exist before!

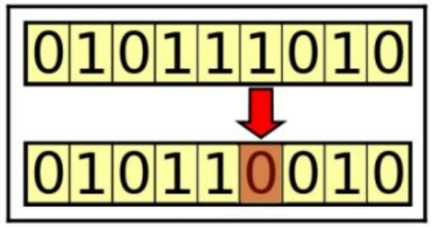
Genetic Algorithm - *Mutation*

- Mutation can be applied to multiple locations within a chromosome

- Mutation can be bitwise changes

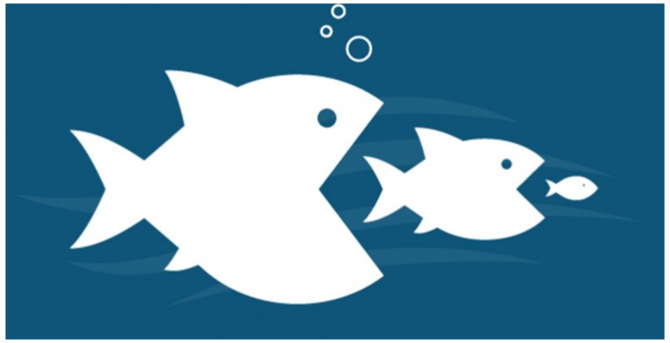


1	203	203
2	100	100
3	403	403
4	2	2
5	4	4
6	167	167
7	800	800
8	34	34
9	55	55
10	38	38
11	232	300
12	19	19
13	432	432
14	21	21
15	1	1
16	0	0
17	22	22
18	82	82
19	88	88
20	418	418
21	87	12
22	2	2
23	43	43
24	11	11
25	5	5
26	24	24
27	39	39
28	50	50
29	8	8
30	12	12



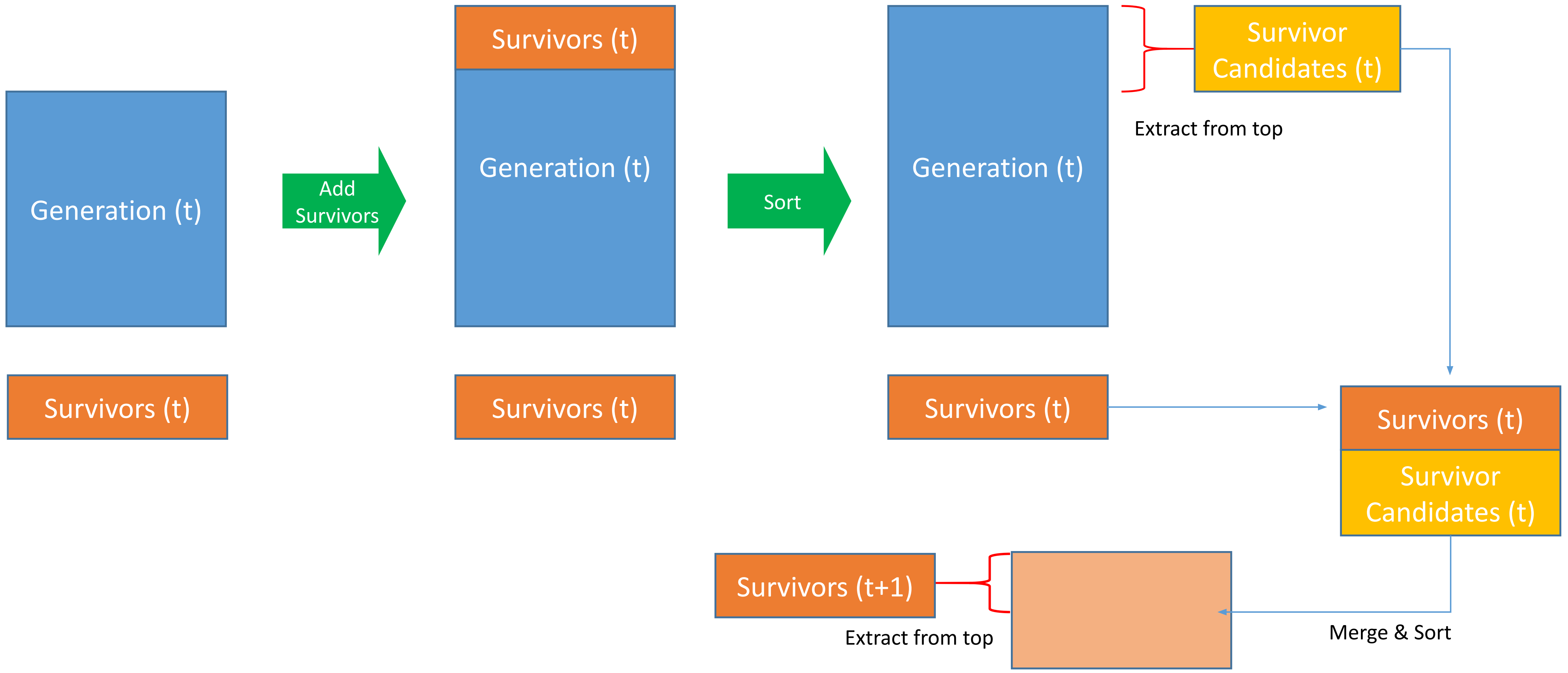
Genetic Algorithm – *Survival of the fittest*

- Individuals are not allowed to move to next generations Only children will go to next generation. Parents die.
- However this may lead to lose better intermediate solutions between generations
- A possible solution:
  - Keep a list of best solutions aside (i.e. Chromosomes with top %10 best fitness score)
  - In each generation, these list of solutions (chromosomes) will join population (merged population will be sorted by fitness score)
  - Keep the list up-to-date with each generation
- This will ensure that all the good solutions will be used throughout the generations
- Convergence of the algorithm improves!



Genetic Algorithm – *Survival of the fittest*

- Keep a separate list of solutions (chromosomes)



Genetic Algorithm – *Use cases*

- Scheduling problems
- Layout design (box, cargo ship, circuit design)
- Feature engineering
- Model hyper-parameter tuning
- Optimization
- Constraint satisfaction
- Self-updating programs/codes
- Music composing

Genetic Algorithm – *Advantages*

- Parallelism
- A larger set of solution space
- Requires less information
- Provides multiple optimal solutions
- Probabilistic in nature, can avoid local min/max
- Genetic representations using chromosomes
- Easy solutions for hard problems, easy to understand concept
- Creativity can be achieved
- Support multi-objective optimization
- Flexible building blocks
- Can work on noisy environments

Genetic Algorithm – *Disadvantages*

- The need for special definitions
- Hyper-parameter tuning
- Computational complexity, can be time-consuming
- Fitness function may be hard to define
- Correct representation can be hard to define
- Convergence not guaranteed