Term Project

Genetic Algorithm for Graph Partitioning

Yoola Choi
2021221291

Contents

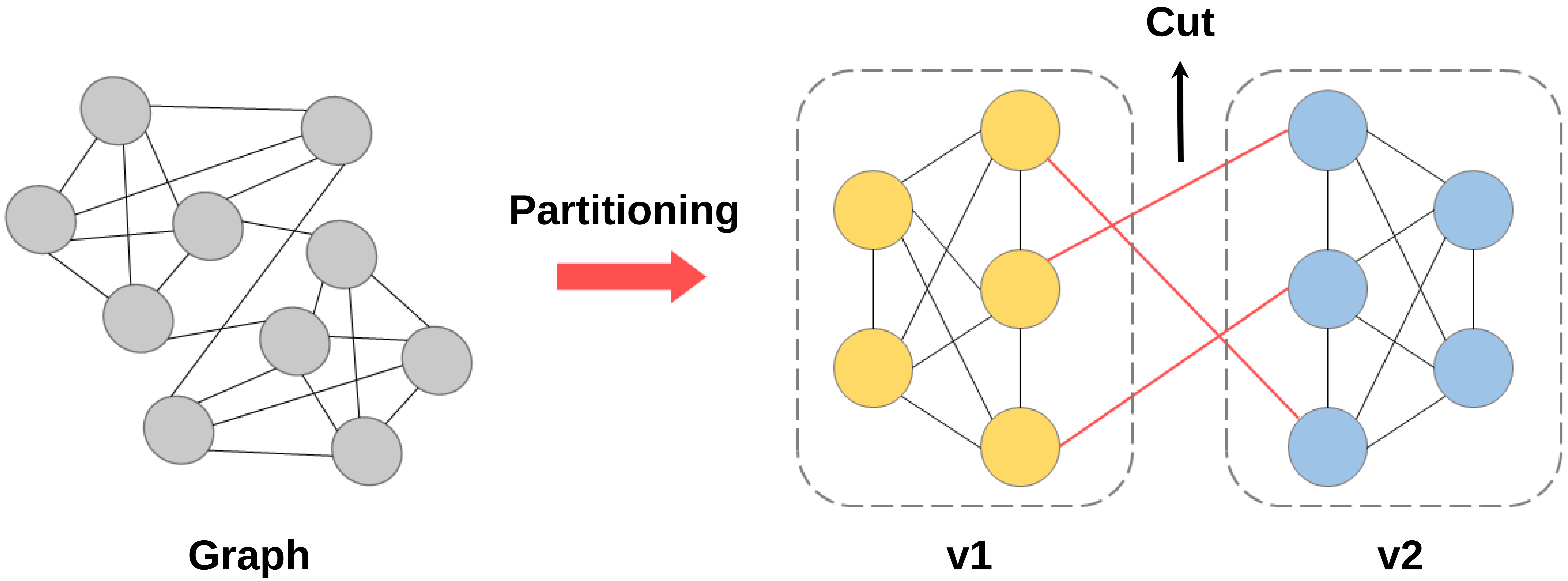
1. Problem Definition
2. Design Decisions
3. Program Execution
4. Limitations and Future Work

Problem Definition

- Graph Partitioning

  Dividing the graph into two disjoint subsets of nodes *v1* and *v2* so that

  - the number of edges between the nodes in the different subsets (cut size) is minimized
  - the sizes of the subsets are equal



Problem Definition

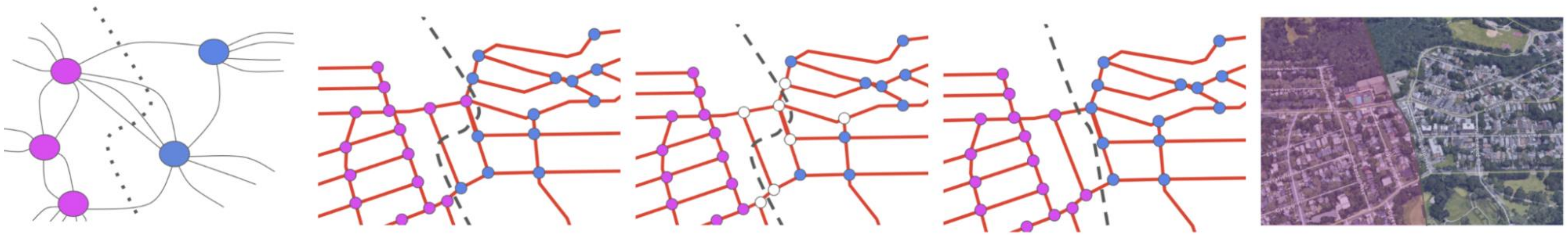
- Why Graph Partitioning is important?

1. Break down a large scale graph problem into smaller subproblems to be solved independently and in parallel → faster processing

2. Many real-world applications (ex. parallel processing, VLSI design)

For example,

- Google maps, where the partitioning algorithm is used to efficiently compute routes.



Cut on the smaller graph. We want to find a similar cut on the original graph.

Mapping it back to the original graph results in a suboptimal cut.

Refinement step is where we find a small region, containing 5% of the nodes, around the cut edges (white nodes).

The minimum cut on this region gives a much better cut.

Corresponding geographical regions.

Problem Definition

- Is Graph Partitioning a combinatorial optimization problem?

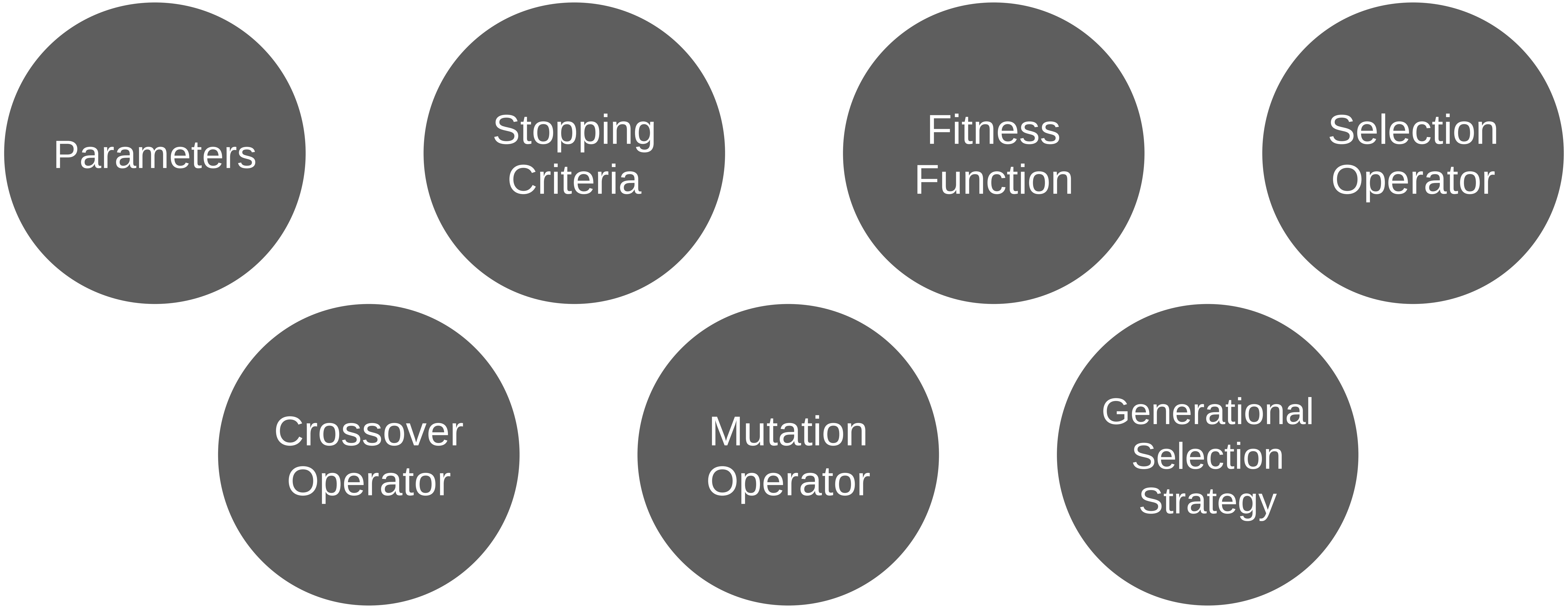
For graph partitioning problem,

- NP-Complete Problem
- To find a optimal solution, each of solutions have to be explored

→ hard to find optimal solution, but it is able to find near optimal solution

→ Combinatorial optimization problem

Design Decisions



Parameters

Stopping Criteria

Fitness Function

Selection Operator

Crossover Operator

Mutation Operator

Generational Selection Strategy

Referred to the paper "Genetic Algorithm and Graph Partitioning." written by Bui, Thang Nguyen, and Byung Ro Moon.

Design Decisions

- Parameters and Stopping Criteria-

POP_SIZE	Initial population size (Integer) → [1, INF)
NUM_NODES	Number of nodes in the graph (Integer), should be an even number → [2, INF)
CONNECT_PROB	Probability to connect two nodes with edge (Float) → (0.0, 1.0]
MUT_PROB	Probability to execute mutation operator (Float) → (0.0, 1.0]
K_IND	Number of individuals for the tournament selection (Integer) → [1, NUM_NODES)
STOPPING_COUNT	Stopping criteria (Integer) → (1, INF)

↳ **If there's no improvement within STOPPING_COUNT times, the program will be terminated**

Design Decisions

- Fitness Function -

Fitness of each individual will be calculated by the equation,

$$F_i = (C_w - C_i) + (C_w - C_b) \,/\, 3$$

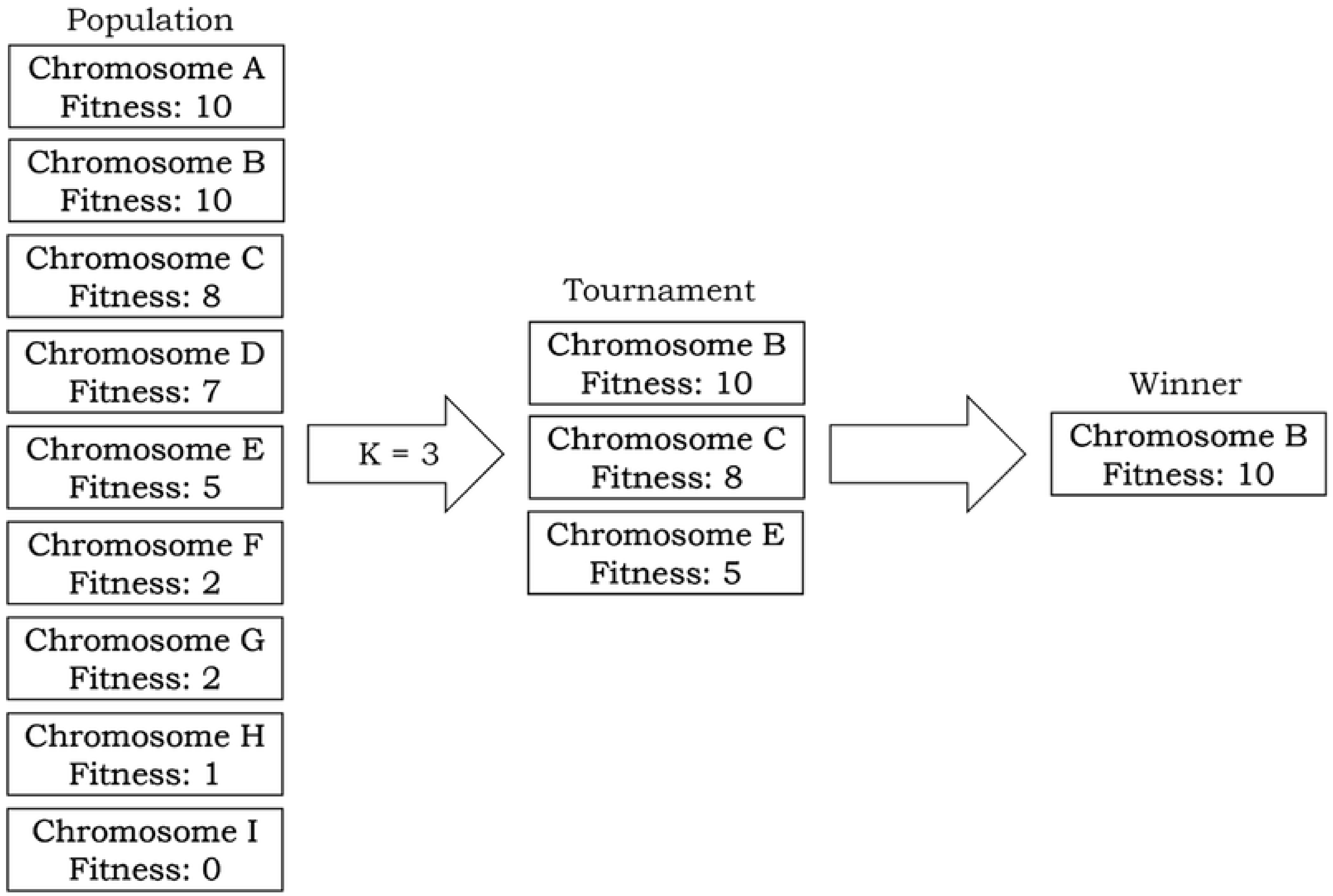
- C_w : Cut size of the worst solution in the population
- C_b : Cut size of the best solution in the population
- C_i : Cut size of solution i

Design Decisions

- Selection Operator -

Tournament selection

- Select K random individuals from the population and pick the best out of them
- Random number K can be adjusted with the parameter, K_IND

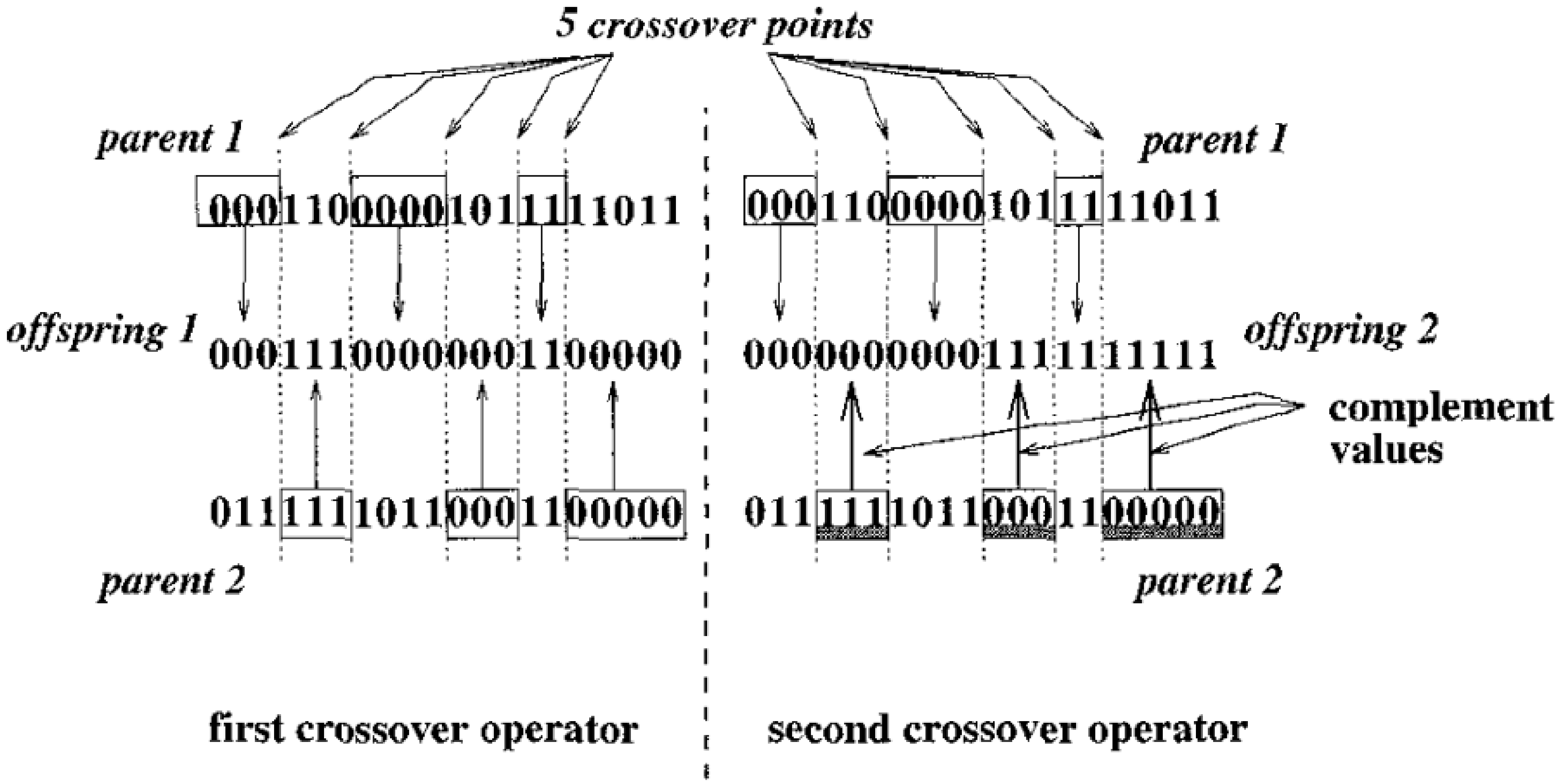


Design Decisions

- Crossover Operator -

Multi-point Crossover

- From the selection operator, two individuals are selected as parents
- 5 cut points are randomly selected
- Offspring 1 and 2 will be generated in different way
- If the partitions of new offspring don't have the same size, the offspring will be discarded

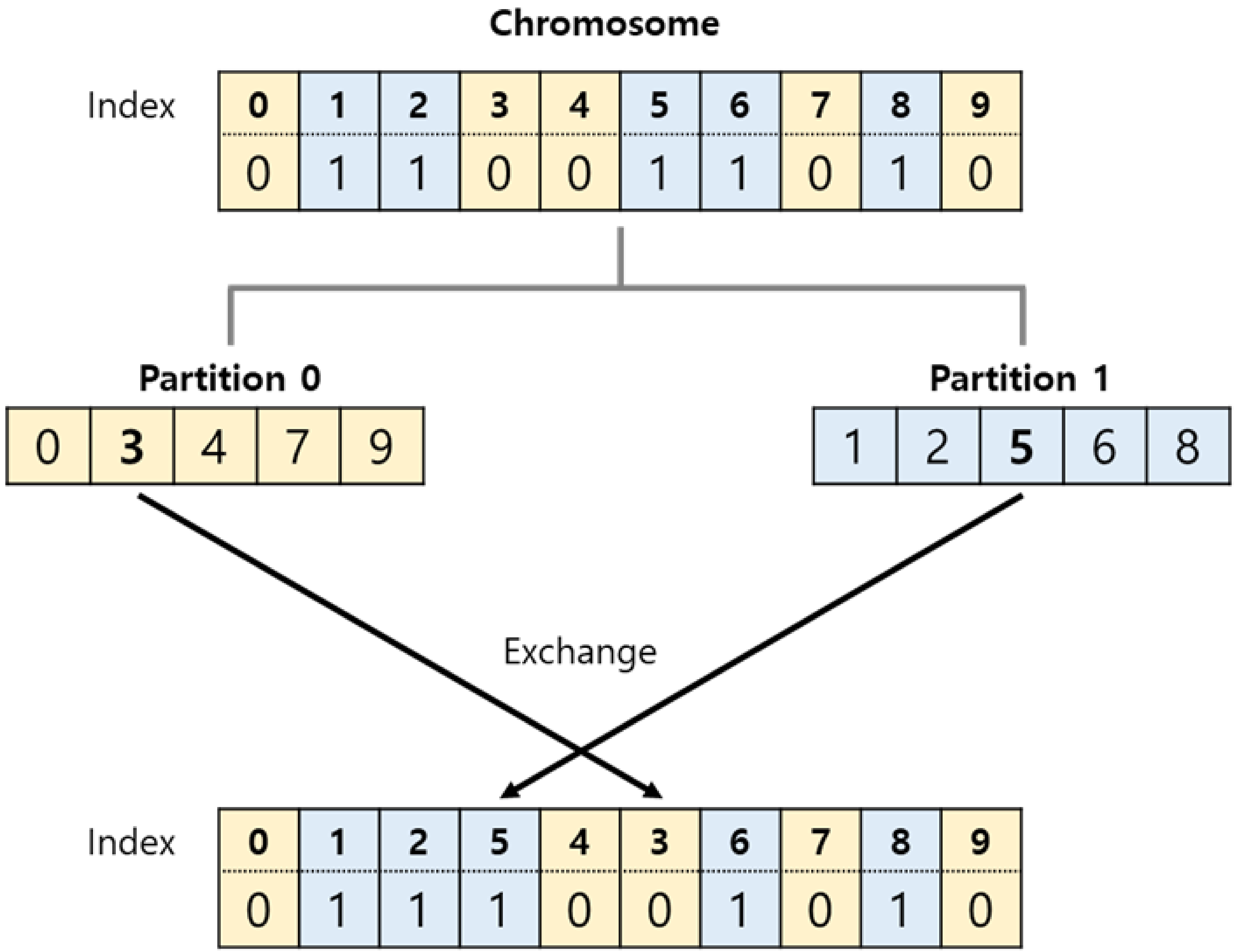


Design Decisions

- Mutation Operator -

Replace one node in a graph with a different, compatible type

→ Select one node from each partitions randomly, and exchange them.



Chromosome

Index	0	1	2	3	4	5	6	7	8	9
	0	1	1	0	0	1	1	0	1	0

Partition 0

0	**3**	4	7	9

Partition 1

1	2	**5**	6	8

Exchange

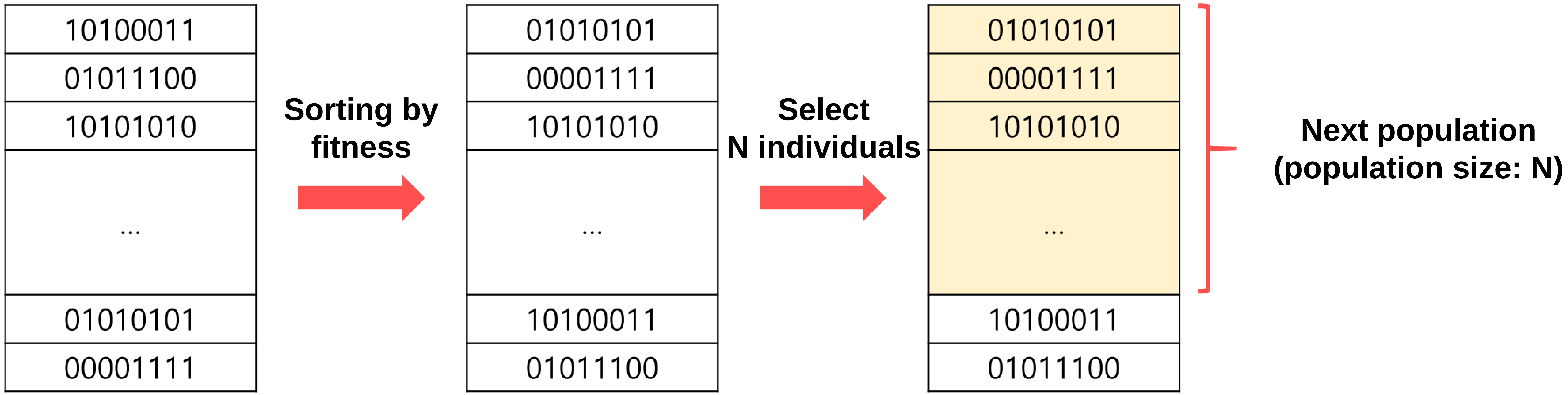
Index	0	1	2	5	4	3	6	7	8	9
	0	1	1	1	0	0	1	0	1	0

Design Decisions

- Generational Selection Strategy -

Elitism

- Keep the best individuals from the parent and offspring population



10100011
01011100
10101010
...
01010101
00001111

Parents and offspring population

Sorting by fitness

01010101
00001111
10101010
...
10100011
01011100

Sorted population

Select N individuals

01010101
00001111
10101010
...
10100011
01011100

Next population (population size: N)

Program Execution

https://github.com/yoooooola/sbse_assignment

Limitations and Future Works

Limitations

1. Time consuming
   - If the number of nodes was more than 300, the program is too slow
2. Crossover operator
   - It generates many offspring that the sizes of both partitions are not equal

Future Works

1. Complement the limitations
2. Expand to k-way partitioning (currently, bisection)

Thank You

Q & A