**Solution of 0-1 Knapsack problem using Genetic Algorithm**

Global variables:

* given from file
  + capacity of knsapsack
  + list of items weights
  + list of items values
* given by user
  + size of population
  + number of generation
  + crossover probability
  + number of crossover points
  + mutation probability
* defined const
  + number of genes
  + number of child
* Step 1 : Start  
  Values from vary inputs are assigned to global variables.

*fetchArgvFromFile* function as argument takes name of file with data and assign them to global variables

*fetchArgvFromUser* function takes data from user input and assign them to global variables

* Step 2 : Generate population

Creating list of genes respectively to weights and values.

Creating individuals – genes randomly selected

* Step 3 : Evolution
  + Parent selection

Measuring fitness of every individual and generate roulette wheel according to fitness values

Random choosing parents from roulette wheel

* + Reproduction

Random choosing number of crossover points

Create child from genes of parents– crossingover

Random mutation – swap two randomly chosen genes

* + Survival selection with elitism

Removing that many individuals with the smallest fitness value as many new children are. Leave individual with the greatest fitness value.

* Step 4 : Terminate

Terminate after given number of generation

Return best solution

**Overview of best solutions and fitness plot**

Data in list means respectively:

- Size of population

- Number of generation

- Crossover probability

- Number of crossover point

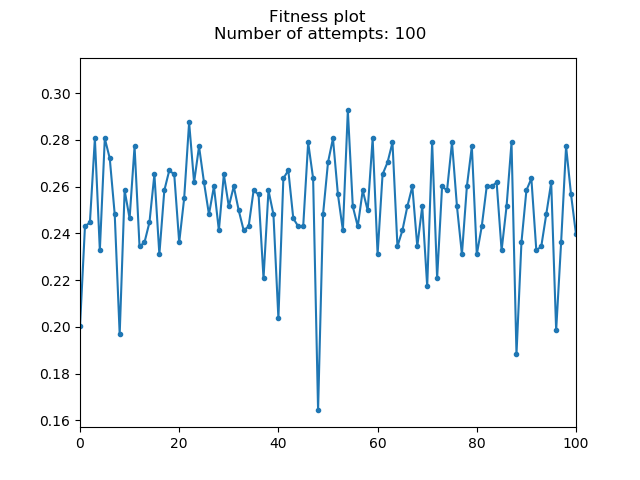
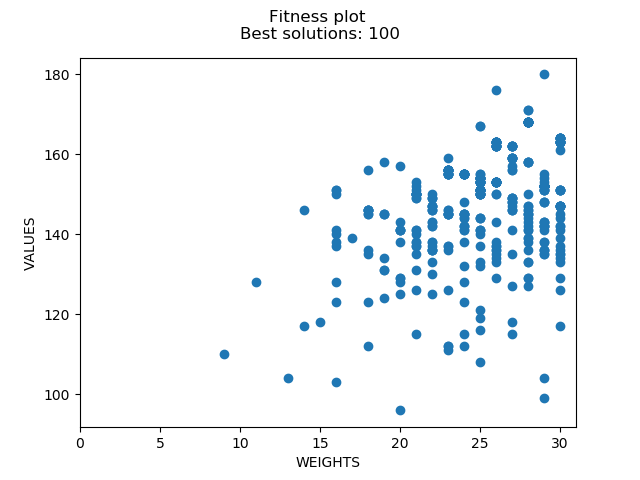
- Mutation probability

[10, 10, 0.9, 1, 0.1]

[50, 10, 0.9, 1, 0.1]

[100, 10, 0.9, 1, 0.1]

[200, 10, 0.9, 1, 0.1]

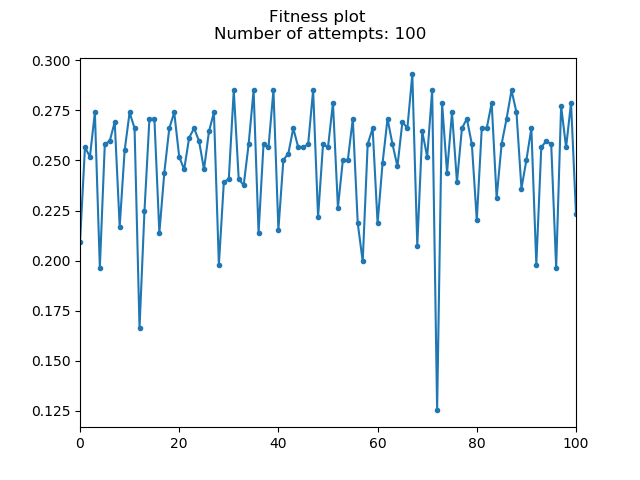
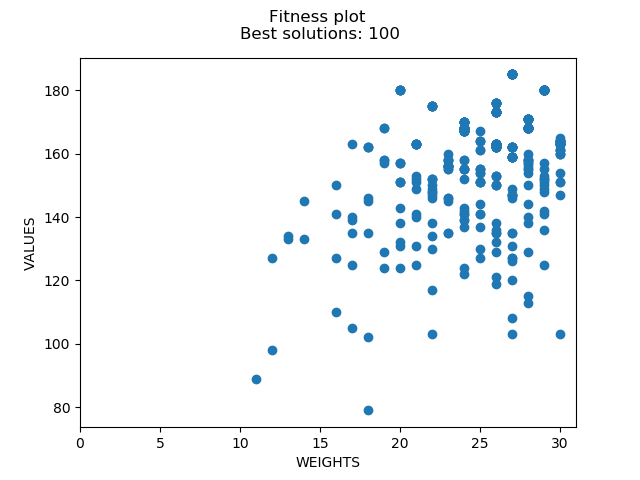


[10, 200, 0.9, 1, 0.1]

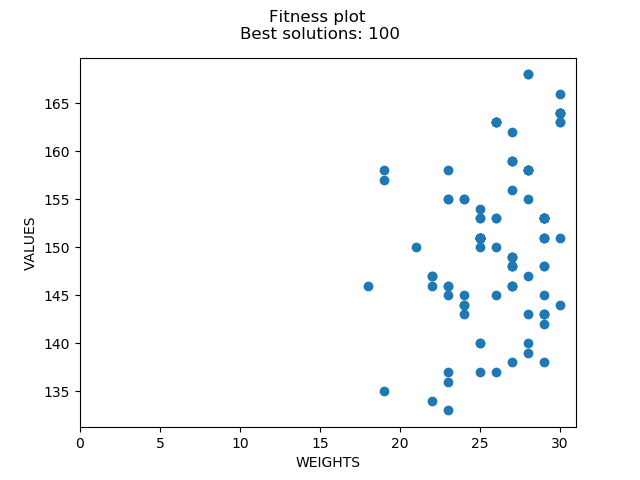
[50, 200, 0.9, 1, 0.1]

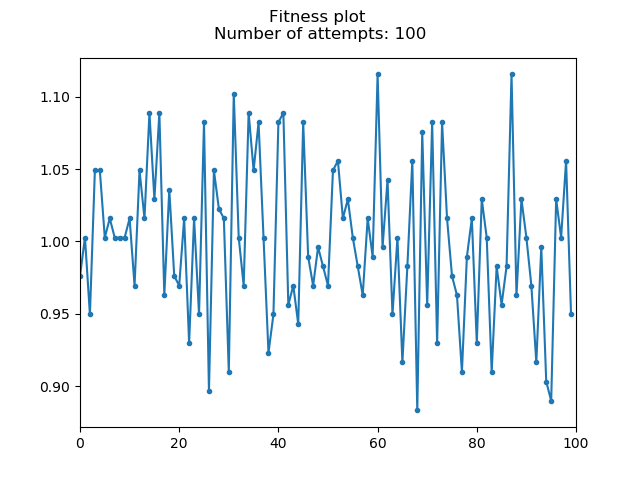
[100, 200, 0.9, 1, 0.1]

[200, 200, 0.9, 1, 0.1]

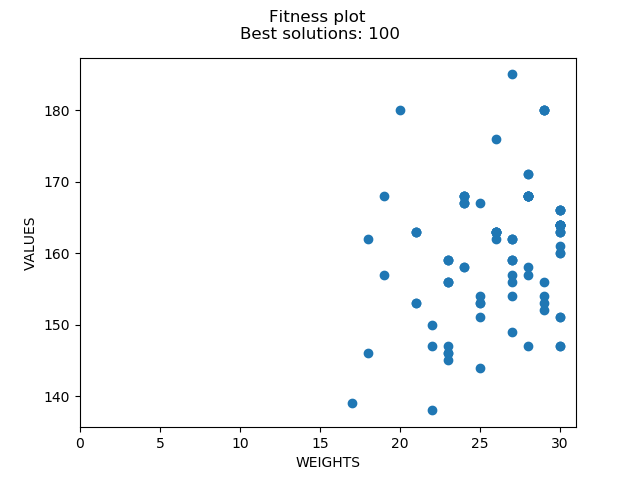


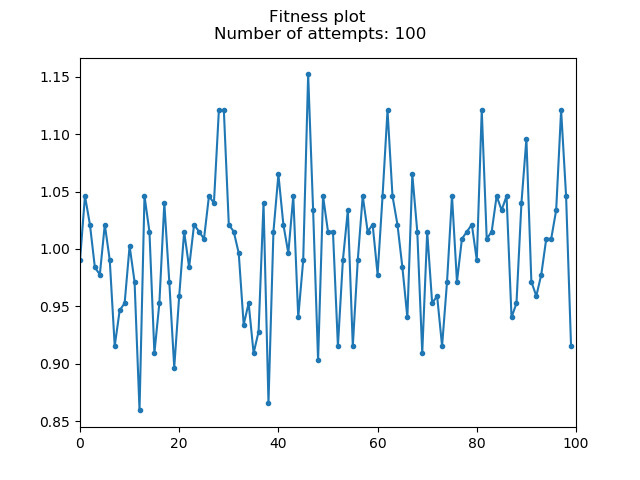
[100, 10, 0.9, 2, 0.1]





[100, 1000, 0.9, 2, 0.1]





**Execution time**

Test data: [10, 10, 0.9, 1, 0.1]

Execution time: 0.001000

Test data: [10, 200, 0.9, 1, 0.1]

Execution time: 0.016000

Test data: [200, 200, 0.9, 1, 0.1]

Execution time: 0.441000

Test data: [100, 100, 0.9, 2, 0.1]

Execution time: 0.096000

Test data: [100, 100, 0.9, 3, 0.1]

Execution time: 0.097000

Test data: [200, 200, 1, 3, 1]

Execution time: 0.470000

Test data: [200, 1000, 1, 3, 1]

Execution time: 2.396000