Genetic Algorithm on Knapsack Problem

Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

First we have **weight[]** and **value[**] in Knapsack which represent the **weight** and **value** for each object.

**Genotype**:

For each individual I randomize 100 bits of **gene** in the int **gene[]**;

My gene[] is set of 1 and 0 which **1** represent we **choose that object and add to our bag**. **0** represent we **do not choose the object**.

For each **individual** we calculate the **fitness** as whatever value it has.

Then we can get, for each individual we calculate fitness by following:

if(gene[i] ==1), weight += weight[i], value += value[i];

Double **fitness = value**;

For the **evolution** part:

The sort function is **Insertion sort**

We sort individuals[] which store individual in population before we start evolution.

Fitter individual means more closer to individuals[0];

1. First we do selection, For each individual in population, we have their **selectionRate** based on **individual fitness/ sum fitness.** Then we do reverse cumulate on the selectionRate, in order to represent “**Fitter individual means more closer to individuals[0]”**, then we set a random number, if it <= cumulate[0] we we more likely to choose individual[0], if random > cumulate[j] and also random <cumulate[j+1] then we more likely to choose cumulate[j+1]
2. Then after choose two Individuals, we crossover them.

We choose two different crossover positions c1, c2. then we crossover where point>c1 and point <c2 from two Individuals, and set them to population if they are not overweight.

1. If we meet mutationRate, then we do mutation on individual1, which we choose 1 single gene position and flip it, 0->1 and 1->0, until we meet that individual not overweight.

After all evolution, we do sort the individual list, to get a ordered population.

In Knapsack, we do 400 generations, until we found the fittest individual in the random populations.

Then we have **conclusion** as following: (Best fitness is 3902.803, with gene[])

INFO: Best Gene Set with value:

3902.803034366157

Best value with ind: 3902.803034366157

1 1/2 0/3 0/4 0/5 0/6 1/7 1/8 0/9 1/10 1

11 0/12 1/13 0/14 1/15 1/16 1/17 1/18 0/19 1/20 1

21 1/22 1/23 1/24 0/25 1/26 1/27 0/28 1/29 1/30 1

31 1/32 0/33 1/34 1/35 1/36 1/37 0/38 1/39 1/40 1

41 1/42 0/43 0/44 1/45 1/46 1/47 1/48 1/49 0/50 1

51 1/52 0/53 1/54 1/55 1/56 1/57 1/58 1/59 1/60 1

61 1/62 1/63 0/64 1/65 0/66 1/67 1/68 1/69 0/70 1

71 1/72 0/73 1/74 1/75 1/76 1/77 1/78 0/79 0/80 0

81 1/82 1/83 1/84 0/85 1/86 1/87 1/88 1/89 1/90 1

91 1/92 1/93 0/94 1/95 1/96 1/97 0/98 0/99 1/100 1

**Pass unit Test:**

