GENETIC ALGORITHM FOR KNAPSACK PROBLEM

Laboratory 1, Introduction to Artificial Intelligence

This is the first **laboratory work** of the **Introduction to Artificial Intelligence** 2020 course in the Computer Science program from **Politechnika Wrocławska** University.

The objective is getting familiar with Genetic Algorithms (GA) meta-heuristics in practice through individual implementation. And test its performance solving the classical *Knapsack Problem*.

The project has been entirely developed in C++ language.

HOW TO COMPILE THE CODE. 🚀

Pre-requisites:

I use `boost::algorithm:split()` to parser lines of *CSV files* easier. So you may need to **install boost library**:

On Debian GNU/Linux based systems:

```
sudo apt-get install libboost-all-dev
```

To compile it simply run:

```
make
```

Or do the same manually:

```
mkdir -p bin/ obj/
g++ -Wall -fexceptions -g -Iinclude -c "src/Lab1.cpp" -o "obj
g++ -Wall -fexceptions -g -Iinclude -c "src/Population.cpp" -
g++ -Wall -fexceptions -g -Iinclude -c "src/Task.cpp" -o "obj
g++  -o "bin/Lab1" "obj/Lab1.o" "obj/Population.o" "obj/Task.
```

PROJECT STRUCTURE

Basically, the code consists of two classes and main function:

Class Population: `Population.h` & `Population.cpp`

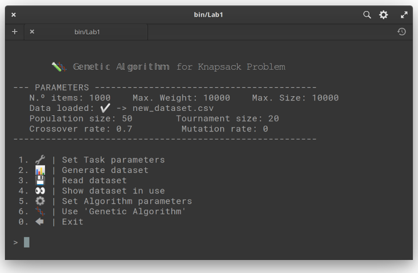
This class is responsible of store individuals population and contains the methods that deal with them. Like Genetic Operators.

Class Task: `Task.h` & `Task.cpp`

This class is responsible of save and manage the data and parameters from our currently problem.

Main function: `main.cpp`

Shows a Main Menu that controls the program from the top.



```
bin/Lab1

     🧪🧬 Genetic Algorithm for Knapsack Problem

--- PARAMETERS ----------------------------------------
   N.º items: 1000    Max. Weight: 10000    Max. Size: 10000
   Data loaded: ✔ -> new_dataset.csv
   Population size: 50        Tournament size: 20
   Crossover rate: 0.7         Mutation rate: 0
-------------------------------------------------------

 1. 🔧 | Set Task parameters
 2. 📊 | Generate dataset
 3. 💾 | Read dataset
 4. 👀 | Show dataset in use
 5. ⚙ | Set Algorithm parameters
 6. 🧬 | Use 'Genetic Algorithm'
 0. 🡄 | Exit

> 
```

Menu is totally working on most Linux Terminal.
On Windows Terminal unicode characters wont be shown properly. (CMD does not support yet.) And to compile it on Windows you will need to change `system("clear")` for `system("cls")` function. Everything else should work exactly the same.
From here you can change the parameters that the data generation or the genetic algorithm will use. can also change this parameters manualy on the main.cpp

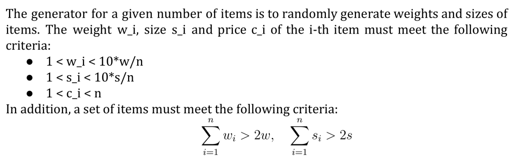
More information is detailed in the source code comments.

SOME DOCUMENTATION NOTES:

Please read first `Lab1.pdf'

1. Criteria Aclaration on Implementation of the task generator

On Knapsack problem
`function generate(n, w, s, output_file)`.
Lab1.pdf sentence that the set of generated items **meets the following criteria:**



The generator for a given number of items is to randomly generate weights and sizes of items. The weight w_i, size s_i and price c_i of the i-th item must meet the following criteria:

- 1 < w_i < 10*w/n
- 1 < s_i < 10*s/n
- 1 < c_i < n

In addition, a set of items must meet the following criteria:

$$\sum_{i=1}^{n} w_i > 2w, \quad \sum_{i=1}^{n} s_i > 2s$$

I was confused here:
What is second criteria for? It's saying I have to program it? Or it's saying it's a property that's fulfilled?

Let's see. Knowing that it is a random distribution, these sums should be approximately "the mean × nº elements"
· e.g for 'w'.

Average (mean) hoped: μ. μ ≈ (1 + (10×w/n))/2 ≈ 5w/n ➔ ∑wi ≈ (5w/n) × n = 5w ≫ 2w

Seeing this we realize that the second criteria is a property that will be met in the vast majority of cases. So we don't have to program anything to force the criteria to be met.

Also, in case there is any doubt, I made tests, checking the value of these sums. The result was what we expected. About 5w.

• • •

Project by: **Nicolás Magro Cruzado** | GitLab ⌨️