COMPUTATIONAL INTELLIGENCE (CI-MAI) - 2016-2017 Project proposal: Genetic Ninjas (packing problems with GAs)

Lluís A. Belanche, David Buchaca, Àngela Nebot

Abstract

Help a group of eighteen ninjas[1] that have been able to mysteriously enter the forbidden city to retrieve some goods. The Chinese empire had stolen precious Japanese treasures which belonged to the ninjas from ancient times.

In order to sneak into the forbidden city each ninja had to bring only a knapsack. Each ninja will try to put in his/her knapsack some of the items that the Chinese army stole. Your goal is to help them choosing which items they should retrieve in order to maximize the value of their own bag. Of course the knapsacks have a finite capacity so they can't retrieve all they would like to ...

1 Introduction

You are provided with 18 records of the items stolen from each of the 18 ninja families. Each ninja has a knapsack with a different capacity. Each ninja is in a different room of the forbidden city, trying to figure out which items he/she has to pick in order to maximize the value without exceeding the capacity of the knapsack. Your task is to maximize the value of items packed into the knapsacks of the ninjas without exceeding their capacities.

2 Data

You are provided with a set of 18 files. Each file contains the data that you should process and work with. These files are organized in the following way:

- The first line in the file contains two numbers.
  - The first integer is the number of items that are in the room.
  - The second integer is the capacity of the knapsack of that ninja
- All the other lines represent an item. Let m be the total number of items.
  - The first number in line $i \in \{2, 3, .., m\}$ is the value of item i.
  - The second number in line $i \in \{2, 3, .., m\}$ is the weight of item i.

This means that the format is as follows:

```
m K
v_1 w_1
```

[1] See `http://en.wikipedia.org/wiki/Ninja`

```
v_2 w_2
 .   .
 .   .
 .   .
v_m w_m
```

Notice that between a value v_i and a weight w_i there is a space.

2.1 File names

The names of the files have the format:

`ninja_X_Y`

- X is the number associated to the family $X \in \{1, 2, \ldots, 18\}$
- Y is the number of items stolen from family X (the same than the first integer value in the first line of the file).

3 Your program

Write a program that takes as input a file ninja_X_Y and outputs the solution of the knapsack for that file. The output of the program has to contain the solution of the knapsack written in two lines, with the following format:

- The first line contains two values: TV and BS separated by a space.
  - TV (total value) is the sum of the items selected to go into the knapsack.
  - BS (best solution) should be 1 if your algorithm found the optimal solution and 0 otherwise.
- The second line is a list of m binary values $p_1, p_2, \ldots, p_m$ separated by spaces. The list has to be created with the convention that $p_i = 1$ means that item i was taken and $p_i = 0$ means otherwise. This line encodes the solution, the format is as follows:

```
TV BS
x_1 x_2 x_3 x_4 x_5 ... x_m
```

4 Goal of the project

Even though this problem might seem quite trivial you will be amazed on how hard it can be. The goal is to solve the problem using genetic algorithms. Be sure to answer the following questions in your report:

Question 1 *Is it a good idea to initialize the chromosomes (individuals) totally at random? If not, how did you do it?*

Question 2 *Try to order the items using their value/weight ratio (from low to high or from high to low). Remember though that the output should be in the same order as the input given (so if the* third *element of the second row in the output is 1 this should mean that item 3 has been taken). Does this improve the performance of the genetic algorithm in terms of time and/or quality of the solution?*

5 Program provided and example

You are provided with a file `solver.py` that already contains some practical code for reading the data and printing the solution. Obviously the file provided won't give you the solution (it prints all '0's). The file contains a `solveIt()` function which is the one that you have to complete. You can, of course, create other functions that might help you solve the problem.

The code must be executable writing the following command (for the example ninja_1_4).

```
python solver.py./data/ninja_1_4
```

For all the other examples it should be analogous. You can try it right away to see that the program provided works.

If you wish you can solve the program using another programing language, provided that it works with the same input and output; you are encouraged to do it in python though.

An example of input/output

The file ninja_1_4 contains the following information:

```
4 11
8 4
10 5
15 8
4 3
```

An output of the problem could be:

```
19 0
0 0 1 1
```

6 Help from magic_d

In `solver.py` there is a function called `magic_d()` that returns the best total value for the knapsack. Some ancient ninjas believe that it is not legal to use it so the code is commented as follows:

```
## MAGIC ##
# best_value = magic_d(weights,values,capacity)
```

You can erase the #s and use the function. Finding the best value can take a while so you can save the best value for each file (ninja_1 up to ninja_18). This function can be useful for detecting when to stop and obviously for assessing the quality of the solution.

7 Final document

In your report, be sure to explain the written code and the obtained results. You should provide what value you obtained for the solution of each of the 18 cases but not the solution itself (the list of ones and zeros). These solutions should be given in a separate .txt file (one for each case).

Remember that in a linux (or OSX) machine you only have to write

```
python solver.py ./data/ninja_1_4  >> sol_1_4
```

to create a file with the output of the program.

You are also expected to send the code with the report. It would be interesting to produce some graphics explaining the time needed to get the result for the different problems.