STN Analysis

Mapping Solutions to Locations

Name of program: hashing.cpp
Authors: Marco Contretas and Gabriela Ochoa

How to use the program

From unix/linux or Mac OS Terminal

- Go to the folder containing the program
- Compile the program

```
$ g++ -Wall -std=c++11 -o Hashing Hashing.cpp
```

- Run the program (example)

```
$ ./Hashing input_data.txt output_data.txt 2
```

In the example above, the program receives an input file (`input_data.txt`) and a numerical parameter (2) that indicates the number of decimal digits to consider. The program produces an output file (`output_data.txt`) that contains the float numbers converted into integers using 2 decimal digits.

In the output file, the fitness values are stored as integers. The solution vectors (with number of components *n*) are stored as integer numbers in hexadecimal, concatenating the vector *n* components.

How the program works

The program converts all the float numbers to integers according to the specified number of decimals. Let us consider an example:

- For a float number
  - *f* = 4.5678
- With *d* = 2 decimals, the integer representation is ,
  - *fint* = 457
- The program converts the float to integer with the following rule
  - *fint* = round(f, d) * 10^d, where *d* is the number of decimals.
- For a solution vector *v* with 3 components (separated with space)
  - *v* = 2.3455 4.5611 3.2345
- With d = 2 decimals, the integer representation is
  - *vint* = 235 456 323
- This will be encoded in hexadecimal with 8 digits
  - *vhex*= 00000eb 00001c8 00000143
- The signature for the node is the concatenation of the hexadecimal numbers
  - 00000eb00001c800000143

Important note

The number converted into integer, should not exceed the maximum size for integer numbers in C++. Tha is, the number *fint* should be in the range -2147483648 <= *fvint* <= 2147483647

Format of input file

The input file should have the following format, and the heading below:

```
Run Value_Current_Best Current_Best n Value_Solution_At_Iteration Solution_At_Iteration
```

Where
`Run`: number of run
`Value_Current_Best`: Fitness value of solution
`Current_Best`: Current best solution vector
`n`: Number of components (dimension) of the solution vectors
`Value_Solution_At_Iteration`: Fitness value of the best next solution
`Solution_At_Iteration`: Next best solution vector

NOTE:
** n is only indicated in the heading, the data items should be separated by a space.

Example of input file with n = 2, and a single run

```
Run Value_Current_Best Current_Best 2 Value_Solution_At_Iteration Solution_At_Iteration
1 19.06153 -17.99794 11.99862 17.31881 -8.99551 10.99452
1 17.31881 -8.99551 10.99452 16.83417 -6.99551 10.99294
```

Output with a single decimal

```
Run Fitness Solution Fitness Next_Solution
1 191 ffffff4c00000078 173 fffffa60000006e
1 173 fffffa60000006e 168 ffffffba0000006e
```

Output with 2 decimals

```
Run Fitness Solution Fitness Next_Solution
1 1906 fffff8f8000004b0 1732 fffffc7c0000044b
1 1732 fffffc7c0000044b 1683 fffffd440000044b
```