Artificial Intelligence Methods (COMP2051 or AE2AIM)

Prof. Ruibin Bai Spring 2019

Coursework: Memetic Algorithm for Multi-Knapsack Problem

1. Introduction

Multi-dimensional knapsack problem is a classic NP-Hard combinatorial optimisition problem used to test the performance metaheuristics. In this coursework, you are asked to write a C/C++ program to solve this problem using a **memetic algorithm** (a variant of genetic algorithm). In addition to submitting source code, a report (no more than 2000 words and 6 pages) is required to describe the algorithm, the experimental results, discussions and reflections on results and performance of the algorithm. **This coursework carries 50% of the module marks**. The rest of module marks comes from the final written exam.

2. Multi-dimensional Knapsack Problem

Multi-dimensional knapsack problem is an extension of the 1D knapsack problem by adding capacity constraints in multiple dimensions. The problem can be formally defined as follows. Given a set of n items numbered from 1 up to n, each with a size vector $\mathbf{v}=(v_{1j}, v_{2j}, v_{3j}, \ldots, v_{mj})$ where v_{ij} is the i-th dimensional size of item j. b_i is the i-th dimensional size of knapsack. p_j is the profit of item j if it is included in the knapsack. Denote x_j be the binary variables to indicate whether item j is included in the knapsack (=1) or not (=0). The problem to be solved is then formulated as follows

$$\sum_{j=1}^{n} p_j x_j$$

Subject to:

$$\sum_{j=1}^{n} v_{ij} x_j \leq b_i \qquad i = 1, \ldots, m$$

$$x_j = \{0,1\}$$

3. Problem instances

In this lab, you are asked to attempt some more challenging instances from paper: P.C. Chu and J.E.Beasley "A genetic algorithm for the multidimensional knapsack problem", Journal of Heuristics, vol. 4, 1998, pp63-86. All the data files are compressed in `mknap-instances.zip`, downloadable from Moodle. The zip file includes 9 problem instance files (each containing 30 instances), 1 data format file `file-format.txt` and 1best known solution file `best-feasible-slns.txt`.

4. Experiments conditions and requirements

The following requirements should be satisfied by your program:

(1) You are required to submit two files only. The first file should contain all your program source codes. The second file is your report.

(2) Your source code should be properly commented.

(3) Your report should include the details of your algorithm (pseudo-code), a description of the parameter tuning process, the results that your algorithm obtains in comparison with the best results in the literature (i.e. gap% to the best results), a short reflection/discussion on the strengths and weaknesses of GA/MA methods.

(4) Name your program file after your student id. For example, if your student number is 2019560, name your program as 2019560.c (or 2019560.cpp).

(5) Your program should compile without errors using one of the following commands (assuming your student id is 2019560 and your program is named after your id):

```
gcc -std=c99 2019560.c -o 2019560
```

or

```
g++ 2019560.cpp -o 2019560
```

(6) After compilation, your program should be executable using the following command:

```
./2019560 -s data_file -o solution_file -t max_time
```

where `2019560` is the executable file of your program, `data_file` is one of problem instance files specified in Section 3. `max_time` is the maximum time permitted for a single run of your MA algorithm. `soluton_file` is the file for output the best solutions by your MA algorithm. The format should be as follows:

```
# of problems
objective value of instance 1
x1 x2 x3 … x_n
objective value of instance 2
x1 x2 x3 … x_n
```

```
… …
objective value of last instance
x1 x2 x3 … x_n
```

An example solution file for problem data file "**mknapcb1.txt**" is available on moodle.

(7) The solution file that your algorithm (`solution_file`) is expected to pass a solution checking test successfully using the following command:

```
./mk_checker -s problem_file -c solution_file
```

where problem_file is one of problem data files in Section 3. If your solution file format is correct, you should get the following command line message "**All solutions are feasible with correct objective values.**"

The solution checker can be downloaded from moodle page. The checker is runnable on CS linux server only.

(8) Your program should take no more than 10 min (i.e. 10 min max_time is set as your stopping criteria of your MA algorithm).

5. Marking criteria

- The quality of the experimental results (30%).
- The quality of codes (30%)
- Report (40%)

6. Submission deadline

29th April 2019, 4pm Beijing Time

7. How to submit

Submit both your files to Moodle

Submit your program file also to cslinux (details to be confirmed later).