Artificial Intelligence Methods (COMP2051 or AE2AIM)

Dr. Xinan Chen Spring 2025

Labs 4: Heuristics & Single Point Metaheuristics

1. Introduction

In the following 3 sessions, you shall practice in using single point metaheuritics to solve a multi-dimensional knapsack problem, which is an extended version of the 1D knapsack problem that you have practiced in pervious labs and has proved to be NP-Hard. The problem has found applications in cloud computing, networking and other practical scenarios. Like in the past, you are asked to write a C/C++ program to solve this problem. A sketch program (downloadable from Moodle) is provided for you also to help you focus more on algorithm implementation.

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

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

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 `mknapinstances.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`, which can be used as a reference to check the performance of your algorithm.

4. Task 1 (lab 4)

Implement a basic simulated annealing algorithm for this problem. Like in lab 2, the initial solution can be generated by some greedy heuristics. You are suggested to use Lundy-Mees nonlinear cooling function ($t = t/(1 + \beta t)$) where β is an input parameter to decide how fast you cool the temperature.

Please fix the SA code I provide and discus with your classmates why these modifications are necessary for the SA code to run properly.