ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

Prof. Dr. E. Zitzler

Bioinspired Optimization and Design FS 2009

Project 1: Knapsack Problem - Part I

Issue Date:	February 24th, 2009
Discussion Date:	March 10th (Task 1), March 17th (Task 2)
Submission Date:	March 5th (Task 1), March 12th (Task 2)

Teaching Assistant: Johannes Bader, `baderj@tik.ee.ethz.ch`

General Guidelines:

- The projects can be worked on in groups of maximal three students.
- The solutions have to be submitted by 24:00 on the given due date. The solutions (program code, plots, text, ...) have to be submitted via email; as attachments only **one** PDF, Word or text document is accepted. Please use the subject line `BOD project <x> task <yz>`.
- The solutions will be graded and returned in the following week's exercise lesson.
- In order to fulfill the attestation conditions, at least 40 points need to be reached for each project and 200 points in total for all 4 projects. Tasks which can easily be skipped are marked as supplementary tasks.

Project Description The knapsack problem is a well-known combinatorial optimization problem. Given is a set of items, each of which has a weight and a profit associated with it, and an upper bound for the capacity of the knapsack. The task is to find a subset of items which maximizes the total of the profits in the subset, yet all selected items fit into the knapsack, i.e. the total weight does not exceed the given capacity. In the first project, our aim is to formalize the problem mathematically and to develop simple randomized algorithms to solve it.

Task 1: Modeling and Implementation (40 points)

A general mathematical model of an optimization problem is

$$\arg\max_{x \in X} f(x)$$

where the real-valued function f is called *objective function* and X is called *search space*. To specify a model for the knapsack problem, assume that we have n items and a capacity of c. Each item i has a positive weight value w_i and a positive profit value p_i associated with it. Both values are integers.

a) Describe how you would represent a solution x for the knapsack problem and define the search space X. *(5 Points)*
b) Give an expression for the objective function f. Solutions that exceed the given capacity bound shall receive an objective function value of zero. *(5 Points)*
c) Describe an algorithm to calculate the objective function value of a given solution x in pseudo code. *(5 Points)*
d) Choose a programming language. Write a program that is able to read all problem parameters (number of items, profit and weight values, capacity) from text files[1]. Further, the program shall generate 10000 random solutions and calculate the objective function value of each of them. The program output consists of the best solution encountered and the corresponding objective function value. Run the program on the different problem instances given on the lecture website. *(10 Points)*
e) **Supplementary task:** Extend your program such that it enumerates all possible solutions and prints out the optimal ones. For each problem instance from task d) report the optimal solution(s), their objective function value and the running time. *(15 Points)*

Task 2: Neighborhood Search (60 points)

Many search heuristics make use of the concept of neighborhood. A neighborhood $N(x)$ of a solution x is a subset of solutions which are in some sense "closer" or "more similar" to x than other solutions.

a) Define a parameterized neighborhood function $N_d(x), x \in X, d \in \mathbf{N}$, for the knapsack problem. The parameter d should reflect the size of the neighborhood. A neighborhood function should have the property that for any $x \in X, d < n : N_d(x) \subset N_{d'}(x) \Leftrightarrow d < d'$. Show that this property holds for your function. *(10 Points)*
b) Implement the Randomized Local Search for the knapsack problem. Run the program on the problem instance given on the lecture website for 100000 iterations and 5 different d values. Plot the current best function value versus the elapsed number of iterations for each of the five runs in the same diagram and store the final best solution found x^*. *(15 Points)*
c) **Supplementary task:** We want to measure how the distance from the best solution x^* relates to the difference in objective function value. Create 10000 random solutions from $N_d(x^*)$ for a reasonable maximal distance d. Plot the solutions in a diagram, where the x-axis denotes the distance in objective space $f(x^*) - f(x)$ and the y-axis the distance in the search space according to the chosen neighborhood function, i.e. $min\{d : x \in N_d(x^*)\}$. Describe the plot and discuss what it reveals about the structure of the problem and the applicability of neighborhood search. *(15 Points)*
d) **Supplementary task:** Contrarily to the Randomized Local Search, the Metropolis algorithm accepts a worse solution with a certain fixed probability T. Implement the Metropolis algorithm for the knapsack problem using the neighborhood function from task 2a). Run the program on the problem instance given on the lecture website for 5 different T values and 100000 iterations each. Assume $d = 1$. Plot the current best objective value versus the elapsed number of iterations for all 5 runs in the same diagram. Also plot the current objective value versus the number of iterations. What T value do you recommend and why? *(20 Points)*

[1]The format of the data files as well as the different problem instances are available at the lecture website `http://www.tik.ee.ethz.ch/sop/education/lectures/BOD/`