Project Report **Ninh DO**
CS 170 SID# 25949105
Professors: Garg & Raghavendra Teammates: NONE
TA: Siqi Liu Due Date: 05/02/2017

1 Overview

The problem is NP-hard. Thus, we do not seek the optimal solution but the polynomial runtime approximate solution. In this report, I will discuss the main idea of my solution, the outline of my coding and the runtime analysis.

2 Main Idea

2.1 Phase I:

In phase I, the hard input is the one which favors both the objective and the constraints at the same time. Since we want to select the items with high resale values while keeping the total cost and the total weight under the bounds, we make the high-resale-value items have either the high cost or the high weight or the class with several conflicts, or any combinations of the three just-mentioned factors. Finally, my hard inputs are:

- The higher the resale value the items have, the higher costs and the higher weights they have, and vice versa.
- The higher the resale value the items have, the more conflicting classes they have, and vice versa.
- The higher the resale value the items have, the higher costs, the higher weights and the more conflicting classes they have, and vice versa.

2.2 Phase II:

In phase II, the goal is to maximize the total resale value and remaining money satisfying the cost, weight and conflict constraints. I implemented this phase with two strategies:

- **Strategy 1:** I first sort the item list in the decreasing order of the value of some heuristic functions evaluated for each item. three heuristic functions I used here are:

$$f_1(i) = r \qquad \text{and} \qquad f_2(i) = r - c \qquad \text{and} \qquad f_3(i) = \frac{r^3}{c \cdot w \cdot n}$$

where i is item, r resale value, c cost, w weight, n number of conflicting items of the item i.
Each heuristic function is particularly useful for a number of input sets. For instance, f_1 works with most input, f_2 works better with input 3 (problem3.in), f_3 with input 21 (problem21.in), etc.
Then I pick the items from the item list in this order. Each time I pick an item, I check for the cost and weight constraints and I eliminate the other items whose classes conflict with the class of the selected item. Besides, I put all selected classes in a set and all conflicting classes in a list.

- **Strategy 2:** I improve the total resale value and remaining money (aka objective value) by "swapping classes". That is, I sort class in the decreasing order of number of conflicting classes. Subsequently, I loop through each class of the sorted class list, if this class has items in the sack, I consider to swap all of the items of this class with all of the items of the conflicting classes that yield the most objective value.

3 Outline of Coding

My coding are object-oriented including some classes:

- Class **Item**: its instance contains all information of an item including name, cost, weight, resale value, heuristic value, etc.
- Class **Collection**: its instance includes a list of all items, a dictionary of *class : set of items*, and a dictionary representing the conflicts of classes.
- Class **GargSack**: its instance includes a list of selected items, an instance of collection, all information we need such as total cost, total weight, cost bound, weight bound, objective value, etc.; a set of all classes in the sack and a list of conflicting classes with the classes in the sack. This class also contain the methods to implement two strategies I mentioned above.

4 Runtime Analysis

Reading file and building collection takes $\Theta(N + C \cdot m)$ time, where m is the average number of conflicting classes in each constraint line.

Sorting item takes $\Theta(N \log N)$ time. Picking items according to strategy 1 takes $\Theta(N)$ time.

Improving the sack according to strategy 2 takes $O(N \cdot m \cdot p)$ time, where p is the average number of items in each class.

The total runtime is $O(N \log N + N \cdot m \cdot p + C \cdot m)$ time. The algorithm is lower-bounded by $\Omega(N)$ and up lower-bounded by various factors depending on the input, such as $O(N \log N)$, $O(N \cdot m \cdot p)$, or $O(C \cdot m)$, whichever is dominant.