ALGORITHM ANALYSIS CSE2046 PROJECT #2 REPORT

Prepared by:

Emirkan KARABULUT 150118052

Ramazan KARKİN 150119512

Semir TATLI 150119004

Introduction

In this project, we designed and applied an algorithm approach for the MCKP (Multi-constraint Knapsack Problem). We used python programming language for coding.

Methods

We implemented a kind of greedy algorithm for this problem. The main idea of this approach using the relation between the value and weight. We used arrays for holding the values. This relation is value/weight ratio (weight here is addition of weights for all knapsacks). And according to this ratio for all elements, we sort decreasing order this results in the arrays and get the items to maximize the knapsack capacities with higher value.

In more details, we have arrays for hold the elements of input files. After we find the solutions of value / weight values, we arrange the arrays proportional to those numbers. We get the values according to the maximum values of value/weight ratio. We add weights we until one knapsack does not have enough place for the next value. When we find the item that exceed the full capacity, we are look at next items that can fit in this knapsack. For example, when exceed the capacity at 14th element, we search the next items that after this item. After that we are looking elements after that number until we are being sure knapsack can get no more weight among all numbers.

Optimization

After greedy approach we wanted to improve solution and we decided to apply local search strategy iteratively make small modifications to the solution to improve the total value. After value/weight ratio we add weights to all knapsacks until the first exceed happen. We find the knapsack where the first overflow occurred and what weight cause, we keep this information in a variable. For example, we add the sorted(value/weight) 5th weight to all the knapsacks, an overflow occurs in the 9th knapsack, we learn that the limiting bag is the 9th knapsack.

We again calculate new value/weight (weight here is weights for 9th knapsack) ratio again for this bag, the weights will be added to all knapsacks according to the highest value/weight ratio after the 5th weights, until the weights do not exceed the knapsacks capacity. Therefore, we usually find a better total value than the value found from the kind of greedy algorithm earlier.

After learning which knapsack was restricted by the greedy approach, we also tried to add the remaining sorted maximum values (not value/weight) to the backpack, which added weight until the first overflow occurred. In any case, we also control whether max values summation gives us better solution or not.

We printed best choice of this algorithm to the output text file.

	Greedy approach (not exactly Greedy algorithm)	Greedy approach + Optimization
test1	1078826	1090337
test2	10084	10668
test3	8466	8709
test4	159	679
sample1	139278	139438
sample2	5097	5544
sample3	7155	7675

References

https://www.thinkmind.org/articles/advcomp_2010_4_30_20106.pdf

https://www.cs.toronto.edu/~nisarg/teaching/373f19/slides/373f19-L9-L10.pdf