MIE-PAA
Paul Rondeau
rondepau
November 3, 2014

Homework 2

In this assignment, the task was to solve the common problem: Knapsack Problem, via 3 methods. Three distinct algorithms were created and compared for accuracy and timing. The first was to be a Branch and Bound algorithm designed to prune the total sample space in order to increase efficiency of the brute force algorithm. Because this algorithm is complete, every possible instance of the problem may be tested in a worst case scenario. The second was an algorithm designed to dynamically analyze the data by creating a table storing previously discovered cost values. The algorithm would then reuse answers previously found instead of recomputing them. This is still a complete algorithm since the solution is guaranteed. Lastly, there was an FPTAS algorithm that modified the previous dynamic algorithm in order to look at less combinations, while maintaining a variable amount of error. This was used to compare performance in terms of speed versus correctness.

The Branch and Bound algorithm would recursively go through the list of items and prune out cases in which the addition of all possible items would not result in a better answer that previously found. It would do this by recording a successful knapsack combination, and then recursively going the items and comparing the potential for each recursive step, if the potential was lower that the current best cost than it needn't be taken.

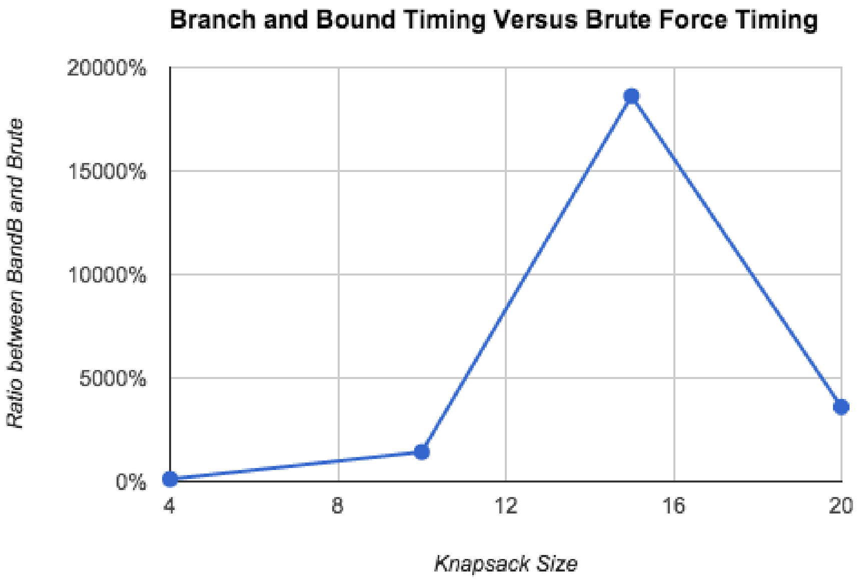
The Dynamic approach utilized an increased sample space while reducing the computation time. For each iteration the algorithm would compute the maximum value that can be attained by using previous calculations. This algorithm runs in pseudo polynomial time.

The FPTAS approach uses the previous dynamic algorithm but applies some preprocessing to reduce the number of calculations needed to be performed. In the algorithm used here, there is a scaling factor Ɛ [0,1] along with the max cost item and the total number of items. Together these form a relative 'bin size', meaning that the costs are all rounded by the following formula:

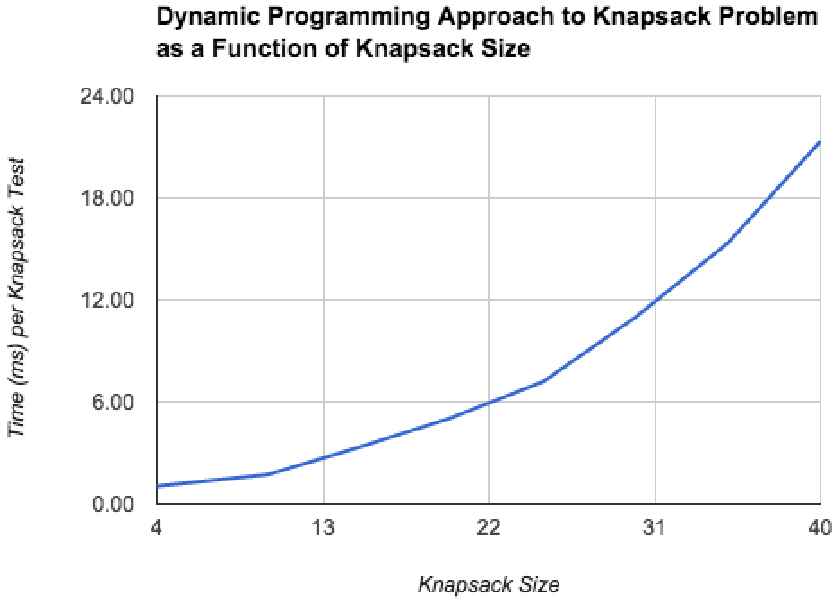
$$K = \frac{\varepsilon * max(c)}{n}$$

This makes the table size required much smaller, thus needed to iterate over a smaller area resulting in a faster algorithm. However, this rounding obviously causes some error.

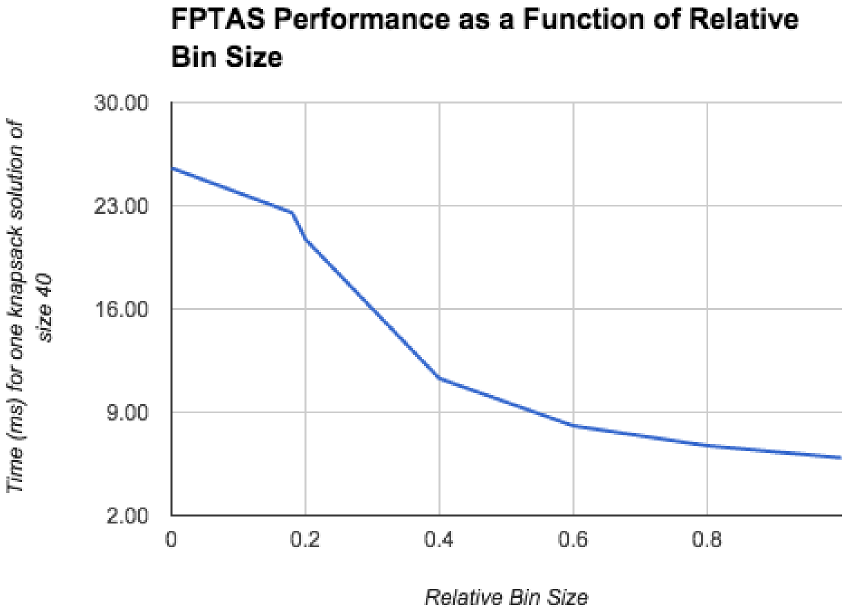
Here is the comparison of the Branch and Bound (BandB) algorithm and the Brute Force (BF) algorithm. Clearly it can be seen the BandB method outperforms the BF in every case testable; however, it is interesting to note the spike at a knapsack of 15, this could be used to the relatively small sample set given where the BandB method is a great choice for algorithms to run on that set of data. However, it is also interesting to note the increase in efficiency for the BandB method as the knapsack size increases.

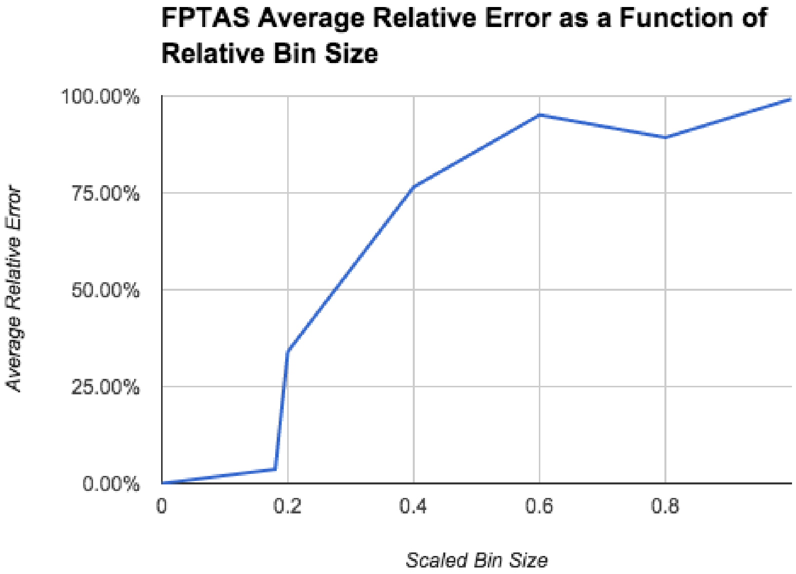


Next is the graph of the Dynamic algorithm performance as a function of knapsack size. It is evident there is an exponential increase in time as the knapsack size increases.

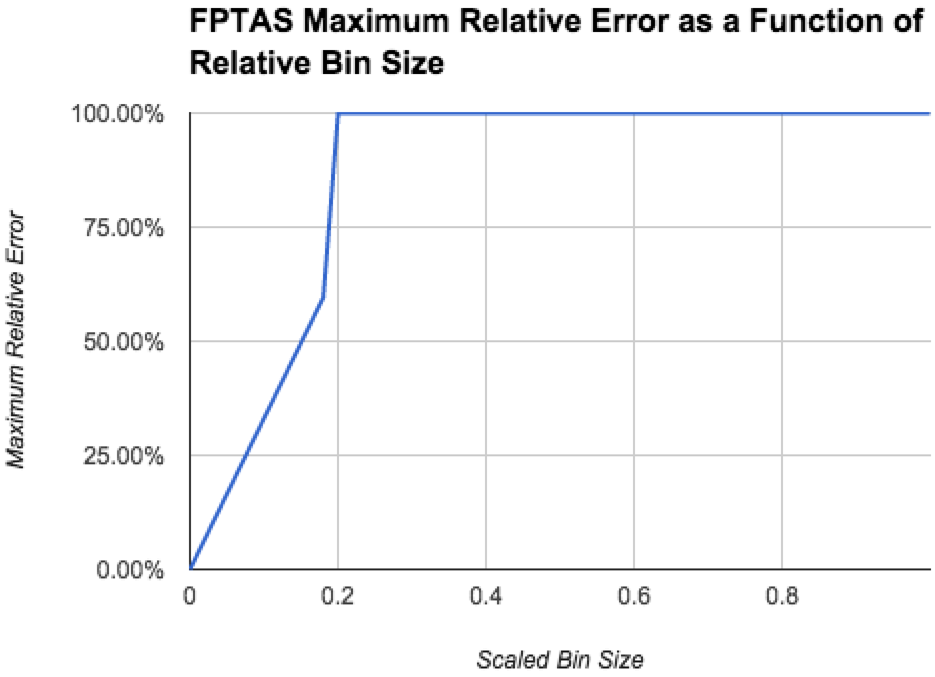


Next is the performance of the FPTAS algorithm as a function of that Ɛ value and subsequently K (the scaling factor). Since Ɛ and K are proportional as Ɛ increases so does K. As a result, when Ɛ is minimized (when K = 1), the error is zero, since no scaling takes place and the runtime remains the same as the dynamic approach. However, as Ɛ increases there is a clear trend between runtime and error. Essentially as Ɛ increases, the execution time is reduced while the average relative error is increased. These trends are illustrated in the graphs below.





There is one surprising metric found when analyzing this data and that was the maximum relative error. As soon as the relative bin size went above 0.2, there was always at least one case of complete error (the cost was calculated as zero).



All calculations included in this report have come from a bash script labeled “run.sh” The values were taken from the average of several trials, and then divided by 50 since there were that many instances in the test files.

In conclusion, depending on the requirements of the algorithm, if the space is limited the Branch and Bound method would be suitable, if time is a constraint than either the Dynamic or the FPTAS algorithm would be suitable depending on the allowed error percentage.

SOURCE CODE:
https://edux.fit.cvut.cz/courses/MI-PAA/_media/en/student/rondepau/assignment2.zip