MIE-PAA
Paul Rondeau
rondepau
November 24, 2014

Homework 3

In this assignment, we were tasked to compare various algorithms' quality and efficiency while changing particular metrics.

The algorithms tested were:

- Brute force
- Branch & Bound
- Dynamic Programming
- Cost/Weight Heuristic

The measurements record were:

- number of algorithm steps
- runtime
- relative error (in case of heuristic)

The variables changed to measure this were:

- instance size (# of items)
- maximum cost
- maximum weight
- knapsack capacity to total weight ratio
- granularity

Brute Force:

The brute force algorithm simple tries every possibility of the knapsack. It is implemented using a binary counter.

Branch & Bound:

The branch & 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.

Dynamic:

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.

Cost/Weight Heuristic:
The greedy cost/weight algorithm used a simple approach, calculate the cost/weight ratio for each element, insert that value into an array, find the max element in this new array and add it to the knapsack.

Experimental Setup:
The setup for this experiment involved creating all of the instances to be tested. I started this this by assigning default values for each variable (close to a good average case). Once the defaults were selected, I would then create instances varying one variable at a time. After all of the instances were created I simply created various scripts that would execute all of the instances for all of the algorithms created in a previous assignments. This script would output timing and error results which I would use to create the tables listed below.

Experiment Results:

Brute Force Algorithm:

BRUTE							
Size	**Time**		**Weight**	**Time**		**Cost**	**Time**
4	45		20	843		20	1008
8	199		40	832		40	861
10	769		100	835		100	823
15	35929		200	807		200	914
			1000	932		1000	1033

Knapsack Capacity to Maximum Weight Ratio	**Time**		**Granularity**	**Time**
0.1	813		0.1	780
0.2	791		0.25	805
0.5	808		0.5	812
0.75	853		0.75	811
0.99	790		0.99	840
2	802		2	769

Greedy Algorithm:

Timing:

GREEDY					
Size	**Time**	**Weight**	**Time**	**Cost**	**Time**
4	40	20	39	20	38
8	42	40	38	40	40
10	41	100	40	100	39
15	48	200	39	200	42
20	45	1000	41	1000	41
40	50				

Knapsack Capacity to Maximum Weight Ratio	Time	Granularity	Time
0.1	46	0.1	42
0.2	40	0.25	38
0.5	45	0.5	38
0.75	39	0.75	41
0.99	35	0.99	39
2	38	2	44

Relative Error:

Greedy Average Relative Error					
Size	**Average Rel Err**	**Weight**	**Average Rel Err**	**Cost**	**Average Rel Err**
4	6.81%	20	5.74%	20	3.54%
8	2.95%	40	4.25%	40	4.16%
10	2.67%	100	2.66%	100	4.53%
15	3.21%	200	3.62%	200	3.91%
20	2.35%	1000	3.29%	1000	4.48%
40	1.12%				

Knapsack Capacity to Maximum Weight Ratio	Average Rel Err	Granularity	Average Rel Err
0.1	12.02%	0.1	2.70%
0.2	6.00%	0.25	3.57%
0.5	5.25%	0.5	4.24%
0.75	1.23%	0.75	3.63%
0.99	0.51%	0.99	3.71%
2	6.12%	2	3.53%

Branch & Bound Algorithm:

BRANCH & BOUND					
Size	**Time**	**Weight**	**Time**	**Cost**	**Time**
4	42	20	65	20	70
8	47	40	68	40	71
10	73	100	68	100	72
15	319	200	68	200	70
20	1802	1000	71	1000	67

Knapsack Capacity to Maximum Weight Ratio	**Time**	**Granularity**	**Time**
0.1	53	0.1	72
0.2	67	0.25	77
0.5	70	0.5	62
0.75	50	0.75	69
0.99	38	0.99	66
2	69	2	63

Dynamic Algorithm:

DYNAMIC					
Size	**Time**	**Weight**	**Time**	**Cost**	**Time**
4	73	20	67	20	171
8	123	40	85	40	148
10	142	100	144	100	146
15	285	200	253	200	145
20	494	1000	1096	1000	168
40	1862				

Knapsack Capacity to Maximum Weight Ratio	**Time**	**Granularity**	**Time**
0.1	56	0.1	143
0.2	75	0.25	146
0.5	150	0.5	139
0.75	205	0.75	143
0.99	267	0.99	140
2	74	2	150

Observations:

Brute Force Algorithm:

As expected, as the size of the number of items increase, the brute force algorithm takes longer to complete as there are an exponential number of combinations to test. It is interesting to note that none of the other factors influenced the brute force complexity very much.

Greedy Algorithm:

As expected the greedy algorithm has a fairly linear increase as the number of items increase, simply due to the fact that there are more ratios to compute. Like the brute force algorithm, the greedy algorithm is also not affected by the other variables changing. There was one additional metric recorded here, average relative error. This value was one of the most interesting observed. As the number of items increase, the error is reduced, which makes sense as there are more items that are being compared against each other. The error also decreases when the maximum weight is increased, but the changing maximum cost has not effect. The biggest contributor to the greedy algorithms' accuracy was the knapsack capacity to maximum weight ratio. As this value approaches 1, the error is almost eliminated. The granularity did not seem to affect the greedy algorithms' relative error.

Branch & Bound Algorithm:

As with the brute force algorithm, the branch & bound algorithm pretty much follows in every way except one. The rate at which the branch & bound algorithm increases when the number of items increase is significantly less than the brute force, enabling us to perform tests on a higher number of items.

Dynamic Algorithm:

The dynamic algorithm results were also quite interesting. As expected, the time to complete problems with more items is increasing. As the maximum weight is increased the dynamic algorithm performs worse, this may be due to the higher variability of the items. The cost fluctuating did not have any impact of performance. The knapsack capacity to maximum weight ratio did affect the performance significantly as well, this time however, as the ratio approaches 1, the performance drops. Lastly, the granularity did not impact the performance at all.

It is quite interesting to observe that the granularity and the maximum cost did not affect any of the algorithms. The main dependencies were on the size, maximum weight and knapsack capacity to maximum weight ratio.

SOURCE CODE:

https://edux.fit.cvut.cz/courses/MI-PAA/_media/en/student/rondepau/assignment3.zip