EECE 7360 Project 4

Garrett Goode and Daniel Hullihen

Spring 2017

1 Introduction

The subset sum problem (also referred to as the “exact knapsack problem”) is defined below.

Let $A = \{a_1, ..., a_n\}$ represent some set of integers. Given a sum s, find a subset $A' \subset A$ such that

$$s = \sum_{i=1}^{n} a_i, for 1 \leq i \leq n.$$

In other words, if we are given a list of numbers and some target sum, we want to find the numbers in the list that would add up to the target sum. Put as a decision problem, the question would be “Is there a subset A’ of A where the sum of the elements of A’ is s?”

In this project, we examined LP and ILP techniques, and then ran our ILP model against our suite of instances.

2 ILP Formulation

The implementation of the ILP formulation we utilized is given by the following AMPL model pseudo-code.

Algorithm 1 AMPL Model for Subset Sum

$values \leftarrow$ {input set}
$X \leftarrow$ [] {empty binary array}

Ensure:
values[i] * X[i] is maximal

Require:
sum(values[i] * X[i]) = target

First parameters are created to store the input data for the problem instance, the set of integers and the target. A second array of binary values is created to track which members of the input set will form the subset that represents the solution.

We elected to maximize the sum of the subset as our objective function, so that even in a situation with no solution we would still be able to achieve the best possible value. This was also the way our greedy algorithm behaved, so it would make it easier to compare the results.

Lastly, our sole condition guaranteed that the sum of the subset would have to be equal to the target. This way, optimal solutions would always stop at the target, despite trying to "maximize" the sum.

3 LP Lower Bound

Unsurprisingly, the LP lower bounds we calculated essentially match the ILP results discussed later in the Results portion of the report. This holds with our expectation, as since subset sum seeks an exact value. One notable difference was the LP models ran much faster than their ILP counterparts, as evidenced by the figures below.

	Bit width																		
Input size	2	4	5	6	8	10	15	17	19	20	21	22	23	24	25	26	27	29	30
2	0.003733	0.004053		0	0.003733	0.003309				0.003551		0.003526		0.00353		0.003753			
4	0.003564	0.003769		0.00361	0.00348	0.003825				0.003753		0.003391		0.003739		0.003628			
6	0.003623	0.003849		0.003805	0.00362	0.003805				0.003529		0.003689		0.003812		0.003554			
8	0.003557	0.003534		0.003704	0.003514	0.004159				0.003676		0.003446		0.003697		0.003459			
10	0.00355	0.003653		0.003658	0.003729	0.003533				0.003616		0.003371		0.003875		0.00382			
16							0.004458	0.003615	0.003728		0.003557		0.004179		0.004216		0.004369	0.003858	
18							0.003701	0.003731	0.003865		0.004061		0.003833		0.00389		0.003782	0.003697	
20							0.003509	0.003844	0.004084		0.004032		0.004056		0.003536		0.00369	0.003629	
22							0.003764	0.003628	0.003923		0.003876		0.003494		0.003605		0.003575	0.00353	
24							0.00363	0.004002	0.003727		0.003491		0.003839		0.003619		0.003323	0.003892	
26							0.003624	0.004089	0.003584		0.003546		0.003628		0.004343		0.003866	0.003517	
28							0.004157	0.003854	0.003678		0.003664		0.004009		0.003784		0.003897	0.00469	
30							0.00385	0.003837	0.003755		0.003627		0.003675		0.003707		0.00366	0.003505	
50							0.003758			0.003782					0.003616				0.003795
60							0.00382			0.003702					0.003567				0.003888
70							0.003873			0.004088					0.003538				0.003664
80							0.003887			0.003989					0.003814				0.003781
90	0.003442		0.003795		0.003616		0.003744			0.003724					0.003901				0.003966
92	0.003598		0.003867		0.004026														
94	0.003465		0.003811		0.003808														
96	0.00393		0.003564		0.003666														
98	0.003614		0.004063		0.003872														
100	0.003707		0.003952		0.00365		0.003749			0.003737					0.003584				0.003859

Figure 1: Runtimes for LP subset sum models

	Bit width																		
Input size	2	4	5	6	8	10	15	17	19	20	21	22	23	24	25	26	27	29	30
2	0	0		0	0	0				0		0		0		0			
4	0.122001	0		0.511278	0.426933	0.436103				0.39752		0.378656		0.156528		0			
6	0.573227	0.344486		0.344632	0.262518	0.347527				0.369383		0.292613		0.046806		0.01049			
8	0.698173	0.787634		0.71189	0.850388	0.40748				0.63521		1.02536		0.138974		0.017803			
10	0.784521	0.582892		0.845672	0.70142	0.670944				45.1428		0.441186		4.62149		0.010194			
16							58.8177	21.4316	57.7484		22.722		46.6254		55.9903		61.7264	0.03662	
18							40.8419	52.244	67.5898		47.3129		54.499		41.1921		52.4214	45.8529	
20							17.8074	44.5947	50.2626		61.144		0.699012		51.3472		51.4473	64.3695	
22							0.72373	59.4118	47.5246		44.5033		49.9935		133.112		42.8144	94.0554	
24							58.782	52.9151	98.5954		49.6893		155.24		96.7399		146.057	57.1084	
26							46.4253	48.3465	96.8108		136.58		140.388		153.763		99.861	211.802	
28							0.941136	44.3117	49.0089		193.401		50.2341		216.754		185.89	198.987	
30							48.739	17.1562	103.806		145.232		125.426		120.239		132.77	117.555	
50							44.3881			96.4059					114.873				80.5586
60							39.9474			0.84924					265.748				150.352
70							1.00856			96.1769					39.7205				253.682
80							0.747665			157.096					225.306				143.687
90	0.976784		0.734031		0.684358		0.745812			105.324					38.6097				31.3719
92	0.670504		0.557017		0.826344														
94	0.850475		0.860876		0.809628														
96	0.729519		0.682609		0.676373														
98	0.341049		0.722678		0.924143														
100	0.780185		0.529281		0.654773		14.0667			124.45					117.322				232.672

Figure 2: Runtimes for ILP subset sum models

4 Results

The results are presented in Figure 3 below. The vertical axis represents the input size and the horizontal axis the word length of the elements in bits.

Input size \ Bit width	2	4	5	6	8	10	15	17	19	20	21	22	23	24	25	26	27	29	30
2	0	0		0	0	0				0		0		0		0			
4	0	0		0	0	0				0		0		0		0			
6	0	0		0	0	0				0		0		0		0			
8	0	0		0	0	0				0		0		0		0			
10	0	0		0	0	0				0		0		0		0			
16							0	0	0		0		0		0		0	0	
18							0	0	0		0		0		0		0	0	
20							0	0	0		0		0		0		0	0	
22							0	0	0		0		0		0		0	0	
24							0	0	0		0		0		0		0	0	
26							0	0	0		0		0		0		0	0	
28							0	0	0		0		0		0		0	0	
30							0	0	0		0		0		0		0	0	
50							0			0					0				0
60							0			0					0				0
70							0			0					0				0
80							0			0					0				0
90	0		0		0		0			0					0				0
92	0		0		0														
94	0		0		0														
96	0		0		0														
98	0		0		0														
100	0		0		0		0			0					0				0

Figure 3: Margin of error for various input size and word length combinations for ILP

Each cell in the above figure is color-coded based on the margin of error the ILP model had in its final solution. This figure is presented mostly to contrast with the other figures in the section, as it is clear the ILP model was able to produce an accurate solution in every instance. Comparing this result with the same figure for the greedy algorithm below, it is abundantly clear that the ILP approach is vastly superior in terms of accuracy. Note that the table in Figure 4 uses an accuracy rating that is the inverse of the ILP table - a rating of 1 is a perfect solution and 0 is instead the worst.

number of elements \ bit width of largest value in set	2	4	5	6	8	10	15	17	19	20	21	22	23	24	25	26	27	29	30
2	1.000	0.167		1.000	1.000	1.000				1.000		1.000		1.000		1.000			
4	1.000	0.941		1.000	1.000	0.649				0.708		0.561		1.000		0.998			
6	1.000	0.750		0.443	0.906	0.855				0.851		0.828		0.895		0.868			
8	1.000	0.917		0.809	0.908	0.744				0.984		0.962		0.947		0.802			
10	1.000	1.000		0.973	0.992	0.994				0.956		0.969		0.962		0.988			
16							0.979	0.995	0.983		0.985		0.950		0.993		0.989	0.933	
18							0.971	0.992	0.991		0.992		0.970		0.993		0.989	0.991	
20							0.996	0.977	1.000		0.969		0.995		0.988		0.987	1.000	
22							0.999	1.000	0.998		0.995		0.971		0.989		0.978	0.997	
24							0.979	0.984	0.992		0.999		0.989		0.990		0.991	0.997	
26							0.979	0.998	0.969		0.994		0.989		0.997		0.988	0.992	
28							0.986	0.997	1.000		0.986		0.996		0.997		0.993	0.991	
30							0.996	0.985	0.995		0.982		0.999		0.998		0.975	0.994	
50							0.998			1.000					0.999				0.997
60							0.994			0.997					1.000				0.603
70							0.999			1.000					0.998				0.997
80							0.998			0.999					0.999				0.984
90	1.000		0.995		0.999		1.000			0.998					0.997				0.995
92	1.000		1.000		0.999														
94	1.000		1.000		1.000														
96	1.000		1.000		1.000														
98	1.000		1.000		0.999														
100	1.000		1.000		0.999		0.997			0.999					1.000				0.980

Figure 4: Margin of error for various input size and word length combinations for the greedy algorithm

Of the 151 instances that were tested, the greedy algorithm was able to correctly solve 28 of them, yielding an 18.5% success rate. Compared to the ILP formulation, which had a 100% success rate, the greedy algorithm is much less successful despite it's lower time complexity. As a reminder, Figure 5 shows the run-times for the exhaustive algorithm.

	2	4	5	6	8	10	15	17	19	20	21	22	23	24	25	26	27	29	30
2	0	0		0	0	0				0		0		0		0			
4	0	0		0	0	0				0		0		0		0			
6	0	0		0	0	0				0		0		0		0			
8	0	0		0	0	0				0		0		0		0			
10	0	0		0	0	0				0		0		0		0			
16							0	0	0		0		0		0		0	0	
18							0	0	0		0		0		0		0	0	
20							0	0	0		0		0		0		0	0	
22							0	0	0		0		0		0		0	0	
24							0	0	1		0		2		0		1	0	
26							0	0	0		1		3		2		6	4	
28							0	0	0		3		6		1		20	27	
30							0	0	1		0		13		34		107	55	
50							402			5					424				
60							600			600					600				
70							600			600					600				
80							600			600					600				
90	600		600		600		600			600					600				29
92	600		600		600														375
94	600		600		600														26
96	600		600		600														600
98	600		600		600														600
100	600		600		600		600			600					600				210

Figure 5: Run time for various input size and word length combinations for the exhaustive algorithm

In general, the amount of margin exhibited by the greedy algorithm seems to come down to how many subsets actually do exist within the set that satisfy the target sum, which depends on factors such as the distribution of integers values in the set and whether a value repeats itself. For example, for sets with several large and small numbers that are similar to each other, there may be more subsets that satisfy a given target sum compared to a set with a more even distribution of integers. Granted, this also depends on the target sum itself. But if there are many subsets that satisfy the target sum, then the greedy algorithm has a better chance of solving a given instance.

When compared to the exhaustive algorithm, the accuracy advantage of the ILP approach is similar to that of the greedy algorithm. However, when the timing results of the ILP algorithm in Figure 2 are compared to that of the exhaustive algorithm, it is actually the much more primitive exhaustive algorithm that has the advantage in solving smaller or less complex instances. This result was surprising, as all other findings seemed to indicate that ILP was more or less a 'magic bullet' for solving our subset sum instances.

5 Conclusion

Overall it seems that ILP is the best approach we have tested to date for solving subset sum instances, in a general sense. More specifically, ILP excels at solving more complex instances that greedy or exhaustive approaches cannot solve in a reasonable amount of time. However, due to its sophisticated approach to generating a solution, it seems to be excessively costly for solving simpler instances. In these instances, one can reach an optimal solution much faster by using a simpler approach. This illustrates that not only is there no perfect method to solve any optimization problem, but it seems that often there is no perfect method for solving every instance of a single optimization problem.