Report for Knapsack solution
by Google OR-Tools

Thanh Phu Bui
Faculty of Software Engineering, University of Information Technology
19522018@gm.uit.edu.vn

30^{th} April 2022

1 Knapsack problem

The knapsack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

The problem often arises in resource allocation where the decision makers have to choose from a set of non-divisible projects or tasks under a fixed budget or time constraint, respectively.

1.1 Problem

You and 4 scientists are researching the radioactive properties of different combinations of elements to find a new clean energy source for the planet. While the radioactive levels in the elements and mixtures are low, you decided that you want to hike out to a remote lab to ensure the safety of the civilian population. Each of you has 1 element packing bag that has the ability to hold **50 pounds (lbs)** of weight and a volume capacity of **50 cubic inches**. In addition, you and the group can only be exposed to maximum radioactivity of **25 rads**, therefore, **each bag can only have a maximum of 5 rads** to ensure safety that no one's bag has too much radiation and is hurting the carrier. Each element has a certain level of value (**utils**) and the goal of the team is to pick the best combination of items that maximize your team's utility while still safely packing the 5 bags. How should the 5 bags be packed with the given item? (**The words "Bag" and "Knapsack" are interchangeable**)

Value	48	30	42	36	22	43	18	24	36	29	30	25	19	41	34	32	27	24	18
Weight (lbs)	10	30	12	22	12	20	9	9	18	20	25	18	7	16	24	21	21	32	9
Volume (in^3)	15	20	18	20	5	12	7	7	24	30	25	20	5	25	19	24	19	14	30
Radioactivity (rads)	3	1	2	3	1	2	0	2	2	1	2	3	4	3	2	3	1	1	3

Styles

Figure 1: Parameters for each of the items.

1.2 Google OR-Tools

OR-Tools is an open source software suite for optimization, tuned for tackling the world's toughest problems in vehicle routing, flows, integer and linear programming, and constraint programming.

I used the knapsack solving tool from Google OR-Tools. Since this is a closed source, then I have fewer things to privately customize. I custom it to stops if there is no optimal solution after 3 minutes.

2 Solving Knapsack with OR-Tools

I use the file *s005.kp* for all of test cases.

2.1 Uncorrelated

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	11922	11921	19780	0.001	**Optimal**
100	28462	28455	42224	0.0004	**Optimal**
200	47601	47601	86024	0.0006	**Optimal**
500	120137	120129	202400	0.0021	**Optimal**
1000	243749	243745	403106	0.0043	**Optimal**

2.2 WeaklyCorrelated

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	12719	12715	14125	0.0004	**Optimal**
100	24642	24637	27933	0.0006	**Optimal**
200	53104	53102	57753	0.001	**Optimal**
500	123199	123199	135595	0.0017	**Optimal**
1000	243336	243336	268214	0.0032	**Optimal**

2.3 StronglyCorrelated

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	12719	12719	16219	0.029	**Optimal**
100	24642	24642	31842	0.0039	**Optimal**
200	53104	53104	66904	0.04	**Optimal**
500	123199	122984	158184	180.03	**Non Optimal**
1000	243336	243169	313969	205.09	**Non Optimal**

2.4 InverseStronglyCorrelated

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	15194	15194	13594	21.538	**Optimal**
100	29592	29592	26592	0.016	**Optimal**
200	63005	62977	56477	12.935	**Optimal**
500	147951	147951	132251	27.083	**Optimal**
1000	292841	292748	261648	186.66	**Non Optimal**

2.5 AlmostStronglyCorrelated

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	12719	12719	16204	18.063	**Optimal**
100	24642	24641	31891	0.0028	**Optimal**
200	53104	53104	66890	0.0024	**Optimal**
500	123199	123134	158328	180.01	**Non Optimal**
1000	243336	243315	314138	203.40	**Non Optimal**

2.6 SubsetSum

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	12719	12719	12719	22.383	**Optimal**
100	24642	24642	24642	0.001	**Optimal**
200	53104	53104	53104	0.0008	**Optimal**
500	123199	123199	123199	0.0012	**Optimal**
1000	243336	243336	243336	0.003	**Optimal**

2.7 UncorrelatedWithSimilarWeights

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	2476436	2401442	19503	0.006	**Optimal**
100	4953340	4902512	39025	0.004	**Optimal**
200	9905740	9904796	79002	0.0009	**Optimal**
500	24764470	24711907	186275	180.01	**Non Optimal**
1000	49529307	49523855	369485	18.84	**Optimal**

2.8 SpannerUncorrelated

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	9808	9808	7728	2.89	**Optimal**
100	21063	21022	16562	189.78	**Non Optimal**
200	41847	41847	32977	307.73	**Non Optimal**
500	104142	104125	82075	272.35	**Non Optimal**
1000	210868	210827	166173	180.07	**Non Optimal**

2.9 SpannerWeaklyCorrelated

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	7688	7683	18104	58.44	**Optimal**
100	16528	16527	38960	178.73	**Non Optimal**
200	32786	32763	77262	244.88	**Non Optimal**
500	81588	81539	192286	235.01	**Non Optimal**
1000	165198	165155	389470	252.18	**Non Optimal**

2.10 SpannerStronglyCorrelated

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	7814	7813	21313	58.29	**Optimal**
100	16600	16592	44492	180.10	**Non Optimal**
200	32802	32761	90261	276.70	**Non Optimal**
500	81687	81672	225472	240.44	**Non Optimal**
1000	165436	165430	457030	246.09	**Non Optimal**

2.11 MultipleStronglyCorrelated

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	12714	12713	20813	35.28	**Optimal**
100	16600	16592	44492	0.011	**Optimal**
200	53100	53100	84200	180.00	**Non Optimal**
500	123198	123198	201698	201.78	**Non Optimal**
1000	243336	243336	399336	23.65	**Optimal**

2.12 ProfitCeiling

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	12719	12718	12708	0.071	**Optimal**
100	24642	24640	24627	1.51	**Optimal**
200	53104	53103	53082	180.53	**Non Optimal**
500	123199	123199	123153	142.92	**Optimal**
1000	243336	243334	243240	201.31	**Non Optimal**

2.13 Circle

No	The maximum weight	Total weight	Total value	Time	Conclusion
50	12719	12719	268001	36.63	**Optimal**
100	24642	24642	519231	0.066	**Optimal**
200	53104	53104	1118954	176.77	**Non Optimal**
500	123199	123199	2595920	96.24	**Optimal**
1000	243336	243336	5127320	180.52	**Non Optimal**

2.14 Conclusion

Results from 13 sets of test cases:

The easy one consists of three groups:

1. **Uncorrelated** and **WeaklyCorrelated** there is no relationship or only a very week relationship between the weight of each item and its value.

2. **SpannerStronglyCorrelated** has the weight of each item which is equal with its value.

The most difficult group is:

1. **SpannerUncorrelated**: In these instances there is no correlation between the profit and weight of an item. Such instances illustrate those situations where it is reasonable to assume that the profit does not depend on the weight. Uncorrelated instances are generally easy to solve, as there is a large variation between the profits and weights, making it easy to fathom numerous variables by upper bound tests or by dominance relations.

2. **SpannerWeaklyCorrelated**: Spanner weakly correlated instances have a very high correlation between the profit and weight of an item. Typically the profit differs from the weight by only a few percent. Such instances are perhaps the most realistic in management, as it is well-known that the return of an investment is generally proportional to the sum invested within some small variations.

3. **SpannerStronglyCorrelated**: The strongly correlated instances are hard to solve for two reasons:

   (a) The instances are ill-conditioned in the sense that there is a large gap between the continuous and integer solution of the problem.

   (b) Sorting the items according to decreasing efficiencies correspond to a sorting according to the weights. Thus for any small interval of the ordered items (i.e. a "core") there ' is a limited variation in the weights, making it difficult to satisfy the capacity constraint with equality.

Today we looked out how to solve the multiple knapsack problems with multiple constraints. When dealing with problems that can be framed linearly, **Google OR Tools** is an excellent resource that offers a wide range of different solvers.