Report for Knapsack solution with Google OR-Tools

Họ tên: Phan Anh Lộc

MSSV: 19521766

I. Experiment set-up

1. Data extraction

Since there is a great deal of data tests and I can’t run through all of them, I created a strategy to randomly get the test cases from the original data set, which is:

• I collected all the folder names and put them together into one array called folder name.

• Then with each folder in the folder array, I changed the name of sub folder into number from 1 to 8 corresponded to its order in the original folder.

• After that, I run through each sub folder from 1 to 8. In each sub folder, I random one number between [0; 1], if that number is greater or equal to 0.5, I would choose the folder R01000, and R10000 if it is less than 0.5

• Finally, I will random one integer in [10; 99] as the test file number, which shares the format as s0**.kp, ** stands for the integer randomized.

2. Google OR-Tools

I used the knapsack solving tool from Google OR-Tools. Since this is a closed source, then I have fewer things to privately customize. Hence, there are only 2 converged conditions in the Google OR-Tools, which are:

- The algorithm is considered as converged if the solver has solved the problem with an optimal solution.
- Besides, it also stops if there is no optimal solution after 3 minutes Output for both Genetic Algorithm and Google OR-Tools session are all kept in the folder output

II. Brief performance about OR-Tools

I might split this section into 3 phases, which are comparison in the event of small input data, medium input data and large input data. The reason behind the data splitting thing is due to the speed of programs when running the data set. In details, there are 3 clear categories of program's performance while running different input data: immediate response, medium reply and low-level speed. Now let us dive into those:

1. **Small data (50 – 500 items)**

Input no.	File link	Items	Max capacity	Total weight	Total value
0	00/1/R01000/s000.kp	50	14778	14721	20995
8	01/1/R10000/s029.kp	50	120877	120655	137103
1	00/2/R01000/s022.kp	100	23122	23122	45031
17	02/2/R10000/s016.kp	100	224429	224428	296428
18	02/3/R10000/s072.kp	200	498025	498025	637025
42	05/3/R01000/s016.kp	200	46559	46559	46559
11	01/4/R01000/s061.kp	500	120268	120263	133549

51	06/4/R10000/s022.k	500	24765240	24713132	180730

s.t. the first number in the file link column is the folder number in the dataset.

As a clear and foremost observation through this table, without a doubt we could realize that OR-Tools gives out a great result. This is since Knapsack's OR Tool gives out optimal solution when it is in the event of small-sized input data

2. **Medium data (1000-2000 items)**

Input no.	File link	Items	Max capacity	Total weight	Total value
52	06/5/R10000/s049.kp	1000	49529110	49523658	367029
44	05/5/R10000/s041.kp	1000	2536287	2536287	2536287
4	00/5/R01000/s007.kp	1000	248176	248175	394138
45	05/6/R10000/s022.kp	2000	4930720	4930720	4930720
53	06/6/R01000/s012.kp	2000	99060009	99050084	746841
93	11/6/R10000/s022.kp	2000	4930720	4930720	4930542

Like Small data, OR-Tools has passed all test cases and given out a great result

3. **Large data (5000-10000 items)**

- Finally, when the input size reaches the huge amount of items, which is 5000 or 10000 items. The difference is shown quite clear when running on my laptop that though solutions are still given from the Knapsack's OR-Tools normally but with just a slower speed. But from test case number 58 until the end of data set, the Knapsack’s OR-Tools solutions barely give out any solution due to the overflow of memory in the processor.

- This problem, fortunately, is solved when I brought these algorithms to Google Colab to do the experiment more effectively without a fear of memory exceeded. Although there are still some of the tests unworkable, the result is still sufficient enough to report my experiment

Input no.	File link	Items	Max capacity	Total weight	Total value
78	09/7/R10000/s028.kp	5000	1505897	1505894	16167894
38	04/7/R10000/s030.kp	5000	12302105	12302105	15828961
54	06/7/R01000/s003.kp	5000	247649308	247622474	1855944
45	09/8/R01000/s039.kp	10000	738271	738264	4121464
55	06/8/R10000/s030.kp	10000	495295590	495246737	3711081
95	11/8/R10000/s083.kp	10000	25018917	25018916	25018011

We can see that OR-Tools didn’t do well when it comes to large input data.

III. Conclusion

- We can see that Google OR-Tools has excelled the DEAP’s simple genetic algorithm framework in all the test cases about the accuracy
- However, there are some big test cases make OR-Tools difficult in solving and there is no solution for those test at all.
- In addition, there is one fact about Knapsack solution from Google OR-Tools is that the huge amount of input data is not the primary factor of which failed the knapsack’s solution tool. It is also about the correlant between numbers in data set. It is a clear observation that in the folder 12, even with

small number of items, e.g. 50 or 100, the solving tool from Google can't handle those.

- Hence, there is a scientific base to say that despite the enormous power from Google, the Knapsack solving's Google-OR Tools by far is neither the best nor the most optimal tool for solving the Knapsack problem. There is many pros and cons to which Google OR-Tools share each other in solving such a problem. We can't tell that which method is exceptionally better than which. The only thing we can tell is that which method is more suitable for the test and which method is less appropriate for the same one.

References

L. Perron and V. Furnon. Or-tools. https://developers.google.com/optimization/