Report for Knapsack solution
with Genetic Algorithm and Google OR-Tools

Tan Ngoc Pham
Faculty of Computer Science, University of Information Technology
19520925@gm.uit.edu.vn

27^{th} March 2021

1 Experiment set-up

1.1 Data extraction

Since there is a great deal of data tests and I can not 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 $R01000$, or else I would choose the rest.
- Finally, I will random one integer in $[10; 99]$ as the test file number, which shares the format as *s0**.kp*, s.t. ** stands for the integer randomized.

1.2 Genetic Algorithm [2]

I did use the source code of *simple Evolution Algorithm* from DEAP framework with some minor changes in order to fit my desire for this experiment.

First of all, about the constants, I have chosen necessary constants as below:

$$\text{POPULATION SIZE} = 100$$
$$\text{P CROSSOVER} = 0.9$$
$$\text{P MUTATION} = 0.3$$
$$\text{MAX GENERATIONS} = \infty$$

I did not choose a limitation for the Genetic Algorithm since I want the algorithm executes until the event of convergence. The convergence conditions are described as follow:

1. If 300 continuous iterations share the same max values or have the max value less than the previous one, then the algorithm is considered to be converged
2. If the algorithm has been executing in 3 minutes without a stop, then the algorithm will end up

1.3 Google OR-Tools [1]

Also, 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:

1. The algorithm is considered as converged if the solver has solved the problem with an optimal solution.
2. 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**

2 Brief performance comparison between GA and OR-Tools

2.1 In general

I chose Google Colab to run 2 these algorithms for its power and non-computer memory needs to execute. Hence, two algorithms have the the same base to compare without a fear of unfairness.

2.2 In details

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:

2.2.1 Small input data

With the size of input data is small, which ranges from 50 items to 500 items. There is a strong evidence that the Google OR-Tools has exceeded the Genetic Algorithm completely in both speed and accuracy aspects.
Let us take a closer look by taking randomly 4 of typical checkpoints, which are 50, 100, 150 and 200 respectively, which have the detailed data below:

Input	File	No.	Max	GA		OR-Tools	
no.	link	of items	capacity	Total weight	Total value	Total weight	Total value
1	00/1/R01000/s085.kp	50	12 847	12 785	19 406	12 818	19 500
40	05/1/R01000/s014.kp	50	11 824	11 824	11 824	11 824	11 824
41	05/2/R01000/s076.kp	100	242 434	242 434	242 434	242 434	242 434
9	01/2/R01000/s072.kp	100	25 041	25 035	27 490	25 037	27 537
3	00/3/R01000/s084.kp	200	125 716	125 664	202 905	125 715	205 566
52	06/4/R01000/s038.kp	500	24 765 380	24 712 416	186 331	24 712 395	187 781

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 surpassing result over Genetic Algorithm. This is since Knapsack's OR Tool gives out optimal solution when it is in the event of small-sized input data.

2.2.2 Medium input data

When the size of input data is around medium level, which are 1000 and 2000 items. One again, we can observe clearly the surpassing power over the Genetic Algorithm from the Knapsack's Google OR-Tools. It has passed all test cases and gives out the more perfect result than Genetic Algorithm does.

Input	File	No.	Max	GA		OR-Tools	
no.	link	of items	capacity	Total weight	Total value	Total weight	Total value
52	06/5/R10000/s084.kp	1000	49 529 018	49 523 477	379 703	49 523 476	381 720
20	02/5/R10000/s069.kp	1000	2 449 426	2 449 317	3 093 317	2 446 433	3 151 433
53	06/6/R01000/s079.kp	2000	99 059 444	99 049 531	711 574	99 048 931	738 207
61	07/6/R10000/s078.kp	2000	4 176 879	4 176 272	3 540 666	4 176 480	3 679 280
21	02/6/R10000/s070.kp	2000	4 936 244	4 936 241	6 194 241	4 933 776	6 341 776

2.2.3 Large input data

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 Genetic Algorithm normally but with just a slower speed. But from test case number 26 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 Colaboratory 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. On my first thought when doing this experiment, I did have a strong feeling of Genetic Algorithm might get a benefit over the Google OR-Tools

when it comes to large input data. However, on the contrary, once a test case is solved by the Google OR-Tools, it would give out more encouraged output than the Genetic Algorithm's despite with a little shift.

For a further understanding, let consider the table:

Input no.	File link	No. of items	Max capacity	GA		OR-Tools	
				Total weight	Total value	Total weight	Total value
22	02/7/R10000/s092.kp	5000	12 252 595	12 252 426	15 062 426	12 249 869	15 769 869
94	11/7/R10000/s068.kp	5000	12485707	12 485 706	12 483 588	12 485 707	12 485 235
79	09/8/R01000/s020.kp	10 000	1 946 495	1 946 468	4 704 868	1 946 474	4 996 174
23	02/8/R10000/s045.kp	10 000	2 483 759	2 483 749	3 010 549	2 483 333	3 186 133
95	11/8/R10000/s048.kp	10 000	25 064 634	25 064 634	25 060 107	25 064 633	25 063 698

2.3 Conclusion

From **2.2.1**, **2.2.2** and **2.2.3**, we can jump into a quick and most foreseeable conclusion of performance between Genetic Algorithm and Google OR-Tools is that Google OR-Tools has excelled the DEAP's simple genetic algorithm framework in all the test cases about the accuracy.

However, one minor thing might make the DEAP's eaSimple still get a position when it is compare between two these methods is that GA did solve and give out the answer to **every single** test case, meanwhile 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 not handle those. But at the same time, the Genetic Algorithm can solve them without a difficulty.

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 both Genetic Algorithm and Google OR-Tools share each other in solving such a problem. We can not 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

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

[2] K. Sastry, D. Goldberg, and G. Kendall. *Genetic Algorithms*, pages 97–125. Springer US, Boston, MA, 2005. ISBN 978-0-387-28356-2. doi: 10.1007/0-387-28356-0_4. URL https://doi.org/10.1007/0-387-28356-0_4.