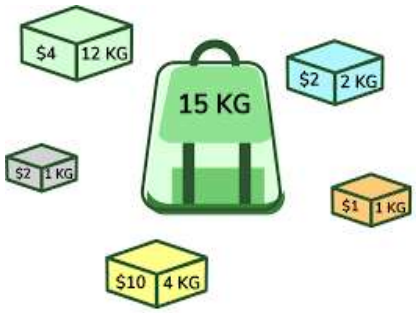
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO

Assignment 2: Solving Knapsack Problems Using Google OR Tools

MÔN: TRÍ TUỆ NHÂN TẠO CS106.M21
GIẢNG VIÊN: TS. LƯƠNG NGỌC HOÀNG
SINH VIÊN THỰC HIỆN: NGÔ ĐỨC VŨ
MÃ SỐ SINH VIÊN: 20520950

I, Tổng quan về bài toán Knapsack



Trong bài toán chiếc túi knapsack, bạn cần đóng gói một tập hợp các mặt hàng, với các giá trị và kích thước đã cho (chẳng hạn như trọng lượng hoặc thể tích), vào một thùng chứa có sức chứa tối đa. Nếu tổng kích thước của các mặt hàng vượt quá sức chứa, bạn không thể đóng gói tất cả. Trong trường hợp đó, vấn đề là chọn một tập hợp con của các mục có tổng giá trị lớn nhất sẽ phù hợp với vùng chứa.
Bằng việc sử công cụ OR-Tools và thuật toán nhánh cận (Branch and bound) cũng một số thư viện khác như os, time, pandas, em đã thực hiện viết chương trình để chạy và lưu kết quả của bài toán chiếc túi knapsack với bộ test case từ kplib, từ đó đưa thống kê và đưa ra một số nhận xét.

II, Thiết lập chương trình và thống kê kết quả bài toán

1, Chọn một mốc chi phí tính toán phù hợp với máy tính

- Cấu hình thiết bị:

Device name	VU
Processor	AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz
Installed RAM	8.00 GB (7.44 GB usable)
Device ID	A12D3571-1462-408C-8450-A63350B3F74D
Product ID	00327-35897-23272-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Xác định số test case để chạy:

- Chạy cho từng bộ kích thước của testcase (**6** bộ là **50,100,200,500,1000,2000**), mỗi nhóm em chạy thử với file 2 test case có tên là: **s000.kp**, **s001.kp**
- Thời gian chạy tối đa mỗi thuật toán là **60** giây.

```
time_limit=60      #Cai dat thoi gian thuc nghiem de danh gia toi uu
```

2, Thiết lập thực nghiệm sao cho OR Tools sẽ dừng khi mà thời gian tính toán cho mỗi lần chạy đã sử dụng hết

- Khai báo biến **time_limit** để xác định thời gian tối đa
- Nếu thời gian chạy xong trên mỗi trường hợp bé hơn thời gian **time_limit** thì kết luận lời giải bài toán là tối ưu (trả về kết quả **True**), và nếu thời gian chạy thuật toán lớn hơn **time_limit** thì mình chưa xác định được kết quả là lời giải bài toán đã tối ưu chưa(trả về kết quả **False**).

```
t0=time.time()         #Thoi gian luc chuan bi chay thuat toan
computed_value = solver.Solve()   #Chuong trinh hoat dong
t1=time.time()-t0    #Bien t1 la hieu thoi gian luc chua chay va sau khi chay
thuat toan
optimal=True         #Tra ve ket qua dung cho bai toan toi uu hay chua
```

```
if t1>=time_limit:   #Neu thoi gian chay > thoi gian toi da, tra ve False
    optimal=False
```

3, Sơ lược về chương trình đã làm để lưu kết quả :

- Tạo hàm lấy dữ liệu từ những file test case ***.kp**

```
def get_data(path):
file_list=open(path,"r").read().split("\n")
capacities=[]
capacities.append(int(file_lit[2]))
values=[]
weights=[[]]
for a in file_list[4:-1]:
b=a.split(" ")
values.append(int(b[0]))
weights[0].append(int(b[1]))
return values, weights, capacities
```

- Tạo các biến dạng list, dictionary để lưu giá trị, chỉ số thư mục.

```
folderpath = "kplib"
dict_group =
{0:"00Uncorrelated",1:"01WeaklyCorrelated",2:"02StronglyCorrelated",3:"03Inver
seStronglyCorrelated",4:"04AlmostStronglyCorrelated",5:"05SubsetSum",6:"06Unco
rrelatedWithSimilarWeights" , 7:"07SpannerUncorrelated"
,8:"08SpannerWeaklyCorrelated" ,9: "09SpannerStronglyCorrelated",
10:"10MultipleStronglyCorrelated", 11:"11ProfitCeiling" ,12: "12Circle"
              }
item_amount_folder={50:"n00050",100:"n00100",200:"n00200",500:"n00500",
                    1000:"n01000",2000:"n02000"}#5000:"n05000",10000:"n10000"}
```

- Tạo các biến dạng dang sách để lưu lại các giá trị best_value, weight, optimal

```
computed_value_table=[]   #bang chua gia tri value tot nhat
total_weight_table=[]  #bang chua gia tri total weight cho loi giai tot nhat
optimal_table=[] #bang danh gia ket qua co toi uu hay khong
list_size=[]     #danh sach chua cac kich thuoc duoc su dung, muc dich de tao
thong tin cot cho table
 list_name=[]     #danh sach ten thu muc duoc su dung, muc dich de tao cot
header cho table
computed_value_list= []   #danh sach gia tri value duoc giai
total_weight=[]    #danh sach gia tri weight tot nhat
list_optimal=[]   #danh sach ket qua the hien loi giai co toi uu hay khong
```

- Nội dung chính hàm main c chương trình:

```
for item in item_amount_folder:
   computed_value_list = []
   total_weight_list=[]
   list_optimal=[]
   list_size.append(item)
    for group in dict_group:
      path=(os.path.join(folderpath,dict_group[group],item_amount_folder[item]
,"R01000","s000.kp"))   #join string to get path to kp file
```

```
      values,weights,capacities= get_data(path)
      solver.Init(values, weights, capacities)
      solver.set_time_limit(time_limit)
      t0=time.time()        #Thoi gian luc chuan bi chay thuat toan
      computed_value = solver.Solve()   #Chuong trinh hoat dong
      t1=time.time()-t0  #Bien t1 la hieu thoi gian luc chua chay va sau khi
chay thuat toan
      optimal=True       #Tra ve ket qua dung cho bai toan toi uu hay chua
      if t1>=time_limit: #Neu thoi gian chay > thoi gian toi da, tra ve False
          optimal=False
      packed_items = []
      packed_weights = []
      total_weight = 0
      #print('Total value =', computed_value)
      for i in range(len(values)):
           if solver.BestSolutionContains(i):
             packed_items.append(i)
             packed_weights.append(weights[0][i])
             total_weight += weights[0][i]
      total_weight_list.append(total_weight)          #them gia tri tong trong
luong vao list
      computed_value_list.append(computed_value)         #them gia tri da duoc
giai trong thoi gian time_limit
      list_optimal.append(optimal)
      print("Result for",path)
      print('Total weight:', total_weight)
      print('Packed items:', packed_items)
      print('Packed_weights:', packed_weights)
                    #them gia tri
   computed_value_table.append(computed_value_list)               #them list
de tao dataframe
   total_weight_table.append(total_weight_list)
   optimal_table.append(list_optimal)
   print("---------------------------------------")
   print("Case with",item, "items done")
   print("---------------------------------------")
```

4. Lập bảng thống kê:

Đoạn code thực hiện xuất kết quả ra file csv:

```
    #Tao dataframe voi value_table, header la list_name
value_table_df=pd.DataFrame(value_table,columns=list_name)
    #Them cot chua kich thuoc
value_table_df.insert(loc=0, column="testcases", value=list_size)
#Ghi ket qua ra file csv
value_table_df.to_csv("value_table.csv", index=False)
#Tuong tu o tren
weight_table_df=pd.DataFrame(weight_table,columns=list_name)
```

```
weight_table_df.insert(loc=0, column="testcases", value=list_size)
weight_table_df.to_csv("weight_table.csv", index=False)
optimal_table_df=pd.DataFrame(optimal_table, columns=list_name)
optimal_table_df.insert(loc=0, column="testcases", value=list_size)
optimal_table_df.to_csv("optimal_table.csv", index=False)
```

- Màn hình terminal khi chạy xong case với kích thước 50 items:

```
Result for kplib\08SpannerWeaklyCorrelated\n00050\R01000\s000.kp
Total weight: 11452
Packed items: [0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 15, 16, 18, 19, 23, 24, 30, 31, 32, 33, 35, 39, 40, 46, 47]
Packed weights: [456, 608, 340, 680, 204, 532, 680, 612, 272, 380, 456, 608, 76, 684, 604, 76, 456, 456, 456, 152, 380, 456, 608, 608, 532]
Result for kplib\09SpannerStronglyCorrelated\n00050\R01000\s000.kp
Total weight: 11532
Packed items: [0, 3, 7, 8, 9, 10, 12, 13, 14, 15, 18, 21, 22, 23, 26, 27, 28, 29, 30, 35, 38, 39, 42, 43, 49]
Packed weights: [380, 340, 680, 612, 272, 612, 340, 680, 340, 204, 612, 272, 136, 204, 408, 136, 544, 612, 680, 476, 680, 612, 612, 612, 476
]
Result for kplib\10MultipleStronglyCorrelated\n00050\R01000\s000.kp
Total weight: 14238
Packed items: [2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 15, 20, 21, 23, 24, 25, 26, 27, 30, 31, 32, 34, 35, 36, 37, 40, 41, 43, 44, 46, 47, 48]
Packed weights: [421, 259, 512, 405, 704, 304, 477, 584, 505, 282, 756, 251, 311, 730, 684, 473, 101, 435, 611, 478, 806, 261, 549, 15, 720,
 399, 2, 494, 244, 326, 192, 568, 239]
Result for kplib\11ProfitCeiling\n00050\R01000\s000.kp
Total weight: 14238
Packed items: [2, 5, 6, 8, 10, 12, 13, 14, 18, 19, 23, 26, 29, 30, 32, 33, 34, 35, 36, 37, 38, 39, 45, 46]
Packed weights: [421, 405, 704, 477, 909, 282, 756, 619, 811, 903, 684, 435, 967, 478, 261, 806, 549, 15, 720, 399, 825, 669, 871, 192]
Result for kplib\12Circle\n00050\R01000\s000.kp
Total weight: 14239
Packed items: [0, 1, 2, 11, 12, 14, 17, 19, 20, 21, 22, 26, 27, 28, 30, 31, 33, 34, 36, 41, 45, 48]
Packed weights: [845, 758, 421, 505, 282, 619, 983, 903, 311, 730, 899, 435, 611, 914, 478, 866, 806, 549, 720, 494, 871, 239]
----------------------------------
Case with 50 items done
----------------------------------
```

- Sau khi chạy hết 6 bộ kích thước, thông qua pandas xuất ra được 3 file, với kết quả: **optimal_table.csv, computed_value_table.csv, total_weight_table.csv**
- Kết quả bảng **total_weight_table** case **s001** và **s002** :

s001.kp	00Uncorre	01Weakly	02Strongl	03Inverse	04Almost	05Subset	06Uncorre	07Spanne	08Spanne	09Spanne	10Multipl	11ProfitCe	12Circle
50	14721	14232	14239	16714	14238	14239	2401482	4569	11452	11532	14238	14238	14239
100	22519	29013	29017	33968	29016	29017	4902253	8748	20824	20956	29016	29015	29017
200	50302	51563	51563	61464	51563	51563	9904900	17274	41116	41288	51558	51562	51563
500	118693	127276	127278	151581	127278	127278	24712055	42898	100076	99928	127278	127277	127278
1000	252480	245972	245968	294834	245972	245972	49525319	84656	198664	198772	245970	245972	245972
2000	502731	498452	498007	597114	498153	498452	99050305	168004	397176	397264	498411	498450	498452

s001.kp	00Uncorre	01Weakly	02Strongl	03Inverse	04Almost	05Subset	06Uncorre	07Spanne	08Spanne	09Spanne	10Multipl	11ProfitCe	12Circle
50	13585	11791	11793	14191	11792	11793	2401186	6578	6437	6461	11790	11792	11793
100	22519	29013	29017	33968	29016	29017	4902253	8748	20824	20956	29016	29015	29017
200	50302	51563	51563	61464	51563	51563	9904900	17274	41116	41288	51558	51562	51563
500	118693	127276	127278	152031	127278	127278	24712055	42898	100016	99928	127278	127277	127278
1000	252480	245972	245972	295477	245972	245972	49525319	84656	198664	198772	245970	245972	245972
2000	493164	497846	497371	596359	497524	497847	99050111	254104	241874	241934	497844	497847	497847

- Kết quả bảng bảng **computed_value_table** case **s001** và **s002**:

s000.kp	00Uncorre	01Weakly	02Strongl	03Inverse	04Almost	05Subset	06Uncorre	07Spanne	08Spanne	09Spanne	10Multipl	11ProfitCe	12Circle
50	20995	15768	17539	14914	17556	14239	19676	13472	10354	28432	21338	14229	300031
100	46537	31064	35617	30468	35611	29017	39791	24228	20550	51656	43316	29001	611418
200	84317	56976	65363	54964	65379	51563	75678	47836	40575	101888	81658	51540	1086483
500	207992	139258	162178	135681	162151	127278	189769	114616	98713	245128	203778	127239	2681868
1000	400811	273052	316368	263434	316415	245972	371246	228624	196050	488672	399170	245877	5182856
2000	808902	552506	638507	534014	638720	498452	748457	459560	391950	979064	806111	498255	10502842

s001.kp	00Uncorre	01Weakly	02Strongl	03Inverse	04Almost	05Subset	06Uncorre	07Spanne	08Spanne	09Spanne	10Multipl	11ProfitCe	12Circle
50	19836	13214	15293	12691	15308	11793	17920	11948	15276	27261	19390	11784	248488
100	46537	31064	35617	30468	35611	29017	39791	24228	20550	51656	43316	29001	611418
200	84317	56976	65363	54964	65385	51563	75678	47836	40575	101888	81658	51540	1086483
500	207992	139258	162178	136031	162154	127278	189769	114616	98700	245128	203778	127239	2681868
1000	400811	273052	316372	263977	316415	245972	371246	228624	196050	488672	399170	245877	5182856
2000	810065	550884	637871	533259	638052	497847	746520	443575	587212	1030634	807944	497640	10490114

- Kết quả bảng **optimal_table** của case **s001** và **s002**:

s000.kp	00Uncorre	01Weakly	02Stronglı	03Inverse	04Almost	05Subset	06Uncorre	07Spanne	08Spanne	09Spanne	10Multipl	11ProfitCe	12Circle
50	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE
100	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE
200	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
500	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE
1000	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE
2000	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
s001.kp	00Uncorre	01Weakly	02Stronglı	03Inverse	04Almost	05Subset	06Uncorre	07Spanne	08Spanne	09Spanne	10Multipl	11ProfitCe	12Circle
50	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
100	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE
200	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
500	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
1000	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
2000	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

5. Kết luận:

Bộ dữ liệu có quá nhiều test case, em đã lựa chọn chiến lược lựa chọn 6 nhóm kích thước **50,100,200,500,1000,2000** của từng nhóm test case từ 0-12. Do cấu hình máy với thời gian chạy từng test case cao nên em chỉ test trên 2 file test key **s000.kp** và **s0001.kp**, nên em có đưa ra một số kết luận như sau:

- Nhóm 12 là nhóm có giá trị lời giải computed_value lớn nhất ở tất cả các kích thước.
- Nhóm 6 có số lượng weight lớn nhất ứng với từng kích thước so với các nhóm còn lại. Lý do sau đó em tìm ra thì do các tệp testcase ở thư mục **"R01000"** trùng với thư mục cùng nhóm 06 là **"R10000"**.

Đối với bảng optimal_table:

- Nhóm 0-1-5 đều đưa ra kết quả **True**, có thể kết luận đây là các nhóm test case dễ.
- Nhóm testcase **8,9,10,11** đưa ra kết quả Optimal là **False** nhiều nhất (**5** lần). Có thể kết luận đây là những kết luận đây là những test case khó.
- Với các kích thước càng lớn dần thì khả năng tìm ra lời giải tối ưu cho bài toán khó hơn, vì vậy ta có thể thấy ở kích thước **50-100** items hầu như tìm ra được lời giải tối ưu. Còn về sau khi kích thước lớn dần thì khả năng tìm được lời giải tối ưu trong thời gian **time_limit** là khó hơn, vì vậy kết quả **False** trả về nhiều hơn.