

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



## **ASSIGNMENT 3**

### **HỌC PHẦN: TRÍ TUỆ NHÂN TẠO GIẢI QUYẾT BÀI TOÁN KNAPSACK SỬ DỤNG GOOGLE OR TOOLS**

<b>Giảng viên hướng dẫn:</b>	Lương Ngọc Hoàng
<b>Lớp:</b>	Trí tuệ nhân tạo - CS106.O22
<b>Sinh viên thực hiện:</b>	Nguyễn Vũ Khai Tâm – 22521293

**TP. Hồ Chí Minh, tháng 4, năm 2023**

# I. TỔNG QUAN

## 1. KNAPSACK PROBLEMS LÀ GÌ ?

- Trong bài toán Ba lô (Knapsack problem), người chơi được cung cấp một tập hợp các vật phẩm, mỗi cái có giá trị và kích thước (hoặc trọng lượng) cụ thể. Mục tiêu là phải chọn lựa một số vật phẩm để đặt vào trong ba lô sao cho tổng giá trị của chúng là cao nhất, nhưng tổng kích thước (hoặc trọng lượng) của các vật phẩm không vượt quá sức chứa của ba lô. Do hạn chế về sức chứa, không phải lúc nào người chơi cũng có thể chứa tất cả các vật phẩm vào trong ba lô, và vì thế, một số vật phẩm có thể sẽ phải được loại bỏ khỏi lựa chọn.



Hình 1: Knapsack problems

- Để giải quyết vấn đề này, trong bài tập dưới đây sử dụng thuật toán Branch and Bound, một kỹ thuật được tích hợp sẵn trong công cụ OR Tools.

## 2. THUẬT TOÁN BRANCH AND BOUND

- Thuật toán Branch and Bound có được coi như phiên bản nâng cấp của Backtracking thông qua cách tiếp cận bài toán bằng việc phân nhánh và giới hạn. Cả hai đều dùng cách chia nhỏ bài toán thành các bài toán con để giải quyết. Tuy nhiên, Branch and Bound nâng cao hiệu quả của Backtracking bằng cách thêm vào bước ước lượng giới hạn cho mỗi bài toán con. Trong khi Backtracking "quay lui" khi gặp điều kiện không thỏa mãn, Branch and Bound "cắt tỉa" các nhánh của cây tìm kiếm bằng cách sử dụng giới hạn để loại bỏ những nhánh không có khả năng dẫn đến giải pháp tối ưu, từ đó tăng tốc độ tìm kiếm. Branch and Bound vừa giữ được tính toàn diện trong tìm kiếm giải pháp tối ưu của Backtracking, vừa tối ưu hóa bằng cách giảm bớt không gian tìm kiếm không cần thiết.

## II. DATASET

- Bộ test instances từ: <https://github.com/likr/kplib>
- Gồm có 13 nhóm Test Instances: 00Uncorrelated, 01WeaklyCorrelated, 02StronglyCorrelated, 03InverseStronglyCorrelated, 04AlmostStronglyCorrelated, 05SubsetSum, 06UncorrelatedWithSimilarWeights, 07SpannerUncorrelated, 08SpannerWeaklyCorrelated, 09Spanner, 10MultipleStronglyCorrelated, 11ProfitCeiling, 12Circle. Mỗi nhóm được thiết kế để kiểm tra khả năng của thuật toán giải quyết bài toán Knapsack dưới các điều kiện và yêu cầu khác nhau, từ việc xử lý các vật phẩm không tương quan cho đến việc giải quyết vấn đề khi có sự tương quan mạnh mẽ hoặc các ràng buộc đặc biệt. Điều này giúp đánh giá tổng quan và toàn diện hiệu suất của thuật toán trên một loạt các trường hợp sử dụng thực tế và giả định.

## III. THỰC HIỆN GIẢI BÀI TOÁN VỚI OR-TOOLS

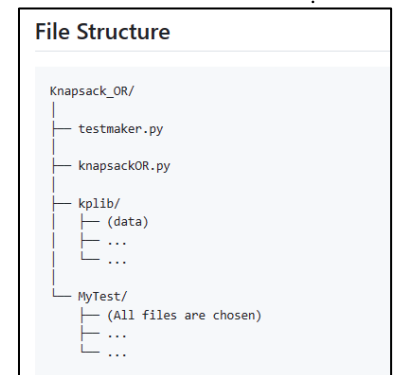
### 1. CHUẨN BỊ BỘ TEST PHÙ HỢP VỚI MÁY TÍNH

- Do giới hạn phần cứng, ở mỗi nhóm, mỗi size và mỗi range em sẽ chỉ lấy một test case. Ví dụ ở nhóm 7, size thứ 50 và ở range R01000 sẽ có test case:

*07SpannerUncorrelated/n00050/R01000/s000.kp*

- Đoạn code dưới đây (hình 3) giúp copy và lấy ra các file trong bộ test từ kplib phù hợp với yêu cầu, sau đó đưa vào folder MyTest (hình 2).

Hình 2: cấu trúc thư mục



```
import os
import shutil

current_path = os.getcwd()

kplib_path = os.path.join(current_path, "kplib")
mytest_path = os.path.join(current_path, "MyTest")

if not os.path.exists(mytest_path):
    os.makedirs(mytest_path)

testgroups = ['n00050', 'n00100', 'n00200', 'n00500', 'n01000']
Rtestgroups = ['R01000', 'R10000']
groupnames = ['00Uncorrelated', '01WeaklyCorrelated', '02StronglyCorrelated', '03InverseStronglyCorrelated',
              '04AlmostStronglyCorrelated', '05SubsetSum', '06UncorrelatedWithSimilarWeights', '07SpannerUncorrelated',
              '08SpannerWeaklyCorrelated', '09SpannerStronglyCorrelated', '10MultipleStronglyCorrelated',
              '11ProfitCeiling', '12Circle']
filenames = ['s000.kp']

for groupname in groupnames:
    for testgroup in testgroups:
        for Rtestgroup in Rtestgroups:
            for suffix in filenames:
                filepath = os.path.join(kplib_path, groupname, testgroup, Rtestgroup, suffix)
                if os.path.exists(filepath):
                    newfilepath = os.path.join(mytest_path, f"{groupname}_{testgroup}_{Rtestgroup}_{suffix}")
                    shutil.copy(filepath, newfilepath)
```

Hình 3: file testmaker.py

## 2. CÀI ĐẶT OR-TOOLS ĐỂ GIẢI QUYẾT CÁC BÀI TOÁN

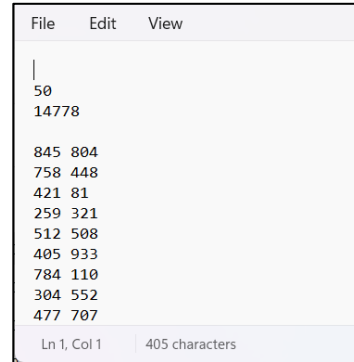
Ta sẽ có những phần như cài đặt thư viện, định nghĩa đường dẫn đến các testcase đã tách ra trong folder MyTest, ghi kết quả vào file csv,... nhưng ở đây sẽ tập trung vào cách lấy thông tin từ file .kp và sử dụng thư viện **ortools**.

- **Đọc và lấy ra thông tin trong file:**

```
if not os.path.exists(filepath):
    continue
capacities = []
values = []
weights = [[]]
with open(filepath, 'r') as f:
    lines = f.read().splitlines()

    capacities.append(int(lines[2]))
    for line in islice(lines, 4, None):
        data = line.split()
        values.append(int(data[0]))
        weights[0].append(int(data[1]))
```

Hình 4: code đọc và lấy thông tin



```
File Edit View
|
50
14778

845 804
758 448
421 81
259 321
512 508
405 933
784 110
304 552
477 707
Ln 1, Col 1 405 characters
```

Hình 5: file .kp

- Khởi tạo ba list rỗng gồm **capacities** để lưu trữ sức chứa của ba lô, **values** để lưu giá trị của từng vật phẩm, và **weights** là một mảng 2 chiều (do OR-Tools solver hỗ trợ bài toán ba lô nhiều chiều - multidimensional knapsack problem), nhưng với bài toán này sẽ chỉ dùng **weight[0]** để lưu trọng lượng của từng vật phẩm. Trong trường hợp này, chỉ có một kích thước ba lô, do đó chỉ có một list trọng lượng bên trong list weights.
- Thêm sức chứa (**lines[2]**) của ba lô vào **capacities**
- Từ dòng thứ 5 (**lines[4]**) trở đi trong file, mỗi dòng gồm hai số, số đầu tiên là giá trị và số thứ hai là trọng lượng của vật phẩm và được thêm vào **values** và **weights[0]** tương ứng.

- **Sử dụng OR-Tools cho bài toán Knapsack Solver:**

- Khởi tạo Solver sử dụng Branch and Bound và gán cho solver, tên là "KnapsackExample".
- Truyền vào solver với **values**, **weights** và **capacities** (đã giải thích các input này ở trên).
- Giới hạn thời gian giải cho **solver.set\_time\_limit()** là 200s.

```
solver = knapsack_solver.KnapsackSolver(
    knapsack_solver.SolverType.KNAPSACK_MULTIDIMENSION_BRANCH_AND_BOUND_SOLVER,
    "KnapsackExample",
)
time_limit = 200
solver.init(values, weights, capacities)
solver.set_time_limit(time_limit)
start = time.perf_counter()
computed_value = solver.solve()
end = time.perf_counter()
elapsed_time = end - start

packed_weights = [weights[0][i] for i in range(len(values)) if solver.best_solution_contains(i)]
total_weight = sum(packed_weights)
isOptimal = "Yes" if solver.is_solution_optimal() else "No"
```

Hình 6: Sử dụng OR-Tools

- **solver.solve():** Sử dụng để thực hiện việc giải quyết bài toán.
- **time.perf\_counter():** Sử dụng để đo thời gian với độ chính xác cao.
- **solver.best\_solution\_contains(i):** Phương thức này kiểm tra xem vật phẩm với chỉ số  $i$  có nằm trong lời giải tối ưu mà solver đã tìm được hay không. Nó được sử dụng để xác định các vật phẩm nào được chọn vào ba lô.
- **solver.is\_solution\_optimal():** Phương thức này trả về True nếu lời giải tìm được là tối ưu, False nếu không hoặc nếu giới hạn thời gian đã được đạt mà không tìm được lời giải tối ưu.

## IV. Thống kê

*Bảng thống kê về lời giải của các bài toán sử dụng OR-Tools*

Group Name	Size	Range	File name	Total Value	Total Weight	Time	isOptimal
00Uncorrelated	n00050	R01000	s000.kp	20995	14721	1.4e-05	Yes
	n00050	R10000	s000.kp	209818	147083	1.5e-05	Yes
	n00100	R01000	s000.kp	46537	22519	2.3e-05	Yes
	n00100	R10000	s000.kp	465424	225212	6e-06	Yes
	n00200	R01000	s000.kp	84317	50302	9.7e-05	Yes
	n00200	R10000	s000.kp	843127	502608	1e-06	Yes
	n00500	R01000	s000.kp	207992	118693	2e-06	Yes
	n00500	R10000	s000.kp	2078639	1185802	2.2e-05	Yes
	n01000	R01000	s000.kp	400811	252480	0.000399	Yes
01WeaklyCorrelated	n01000	R10000	s000.kp	4005798	2522529	0.000398	Yes
	n00050	R01000	s000.kp	15768	14232	0.000163	Yes
	n00050	R10000	s000.kp	157504	142272	0.000203	Yes
	n00100	R01000	s000.kp	31064	29013	0.000143	Yes
	n00100	R10000	s000.kp	310315	289895	0.000309	Yes
	n00200	R01000	s000.kp	56976	51563	0.000267	Yes
	n00200	R10000	s000.kp	569063	515196	0.000619	Yes
	n00500	R01000	s000.kp	139258	127276	0.000632	Yes
	n00500	R10000	s000.kp	1390993	1271685	0.008396	Yes
02StronglyCorrelated	n01000	R01000	s000.kp	273052	245972	0.001895	Yes
	n01000	R10000	s000.kp	2727089	2457528	0.000595	Yes
	n00050	R01000	s000.kp	17539	14239	0.012105	Yes
	n00050	R10000	s000.kp	175283	142283	0.015401	Yes
	n00100	R01000	s000.kp	35617	29017	0.07493	Yes
	n00100	R10000	s000.kp	355961	289961	0.124095	Yes
	n00200	R01000	s000.kp	65363	51563	200.018879	No
	n00200	R10000	s000.kp	653200	515200	200.018977	No
	n00500	R01000	s000.kp	162178	127278	200.039113	No
	n00500	R10000	s000.kp	1620685	1271685	200.024109	No
	n01000	R01000	s000.kp	316372	245972	200.060419	No
	n01000	R10000	s000.kp	3161219	2457219	200.037665	No

<b>03InverseStronglyCorrelated</b>	n00050	R01000	s000.kp	14914	16714	1.409472	Yes
	n00050	R10000	s000.kp	149036	167036	2.675347	Yes
	n00100	R01000	s000.kp	30468	33968	0.000747	Yes
	n00100	R10000	s000.kp	304466	339466	0.000851	Yes
	n00200	R01000	s000.kp	54964	61464	0.005711	Yes
	n00200	R10000	s000.kp	549209	614209	0.013629	Yes
	n00500	R01000	s000.kp	136031	152031	200.120424	No
	n00500	R10000	s000.kp	1356100	1515100	200.03045	No
	n01000	R01000	s000.kp	263977	295477	200.045754	No
	n01000	R10000	s000.kp	2632946	2946946	200.025659	No
<b>04AlmostStronglyCorrelated</b>	n00050	R01000	s000.kp	17556	14238	0.008123	Yes
	n00050	R10000	s000.kp	175434	142280	0.012151	Yes
	n00100	R01000	s000.kp	35611	29016	0.024874	Yes
	n00100	R10000	s000.kp	355902	289960	0.046423	Yes
	n00200	R01000	s000.kp	65385	51563	200.00628	No
	n00200	R10000	s000.kp	653363	515198	200.003392	No
	n00500	R01000	s000.kp	162154	127278	200.014338	No
	n00500	R10000	s000.kp	1620463	1271681	200.01461	No
	n01000	R01000	s000.kp	316415	245972	0.152089	Yes
	n01000	R10000	s000.kp	3161935	2457530	200.050501	No
<b>05SubsetSum</b>	n00050	R01000	s000.kp	14239	14239	9.6e-05	Yes
	n00050	R10000	s000.kp	142283	142283	5.4e-05	Yes
	n00100	R01000	s000.kp	29017	29017	1.9e-05	Yes
	n00100	R10000	s000.kp	289961	289961	0.002105	Yes
	n00200	R01000	s000.kp	51563	51563	2.7e-05	Yes
	n00200	R10000	s000.kp	515200	515200	6.9e-05	Yes
	n00500	R01000	s000.kp	127278	127278	0.000259	Yes
	n00500	R10000	s000.kp	1271685	1271685	0.003684	Yes
	n01000	R01000	s000.kp	245972	245972	0.000316	Yes
	n01000	R10000	s000.kp	2457533	2457533	0.000414	Yes
<b>06UncorrelatedWithSimilarWeights</b>	n00050	R01000	s000.kp	19676	2401482	0.029783	Yes
	n00050	R10000	s000.kp	19676	2401482	0.030852	Yes
	n00100	R01000	s000.kp	39791	4902253	6.835908	Yes
	n00100	R10000	s000.kp	39791	4902253	7.042922	Yes
	n00200	R01000	s000.kp	75678	9904900	5e-06	Yes
	n00200	R10000	s000.kp	75678	9904900	3e-06	Yes
	n00500	R01000	s000.kp	189769	24712055	200.031833	No
	n00500	R10000	s000.kp	189769	24712055	200.02368	No
	n01000	R01000	s000.kp	371246	49525319	0.039496	Yes
	n01000	R10000	s000.kp	371246	49525319	0.033505	Yes
<b>07SpannerUncorrelated</b>	n00050	R01000	s000.kp	13472	4569	0.002256	Yes
	n00050	R10000	s000.kp	135733	46295	0.001929	Yes
	n00100	R01000	s000.kp	24228	8748	200.01381	No
	n00100	R10000	s000.kp	243893	88483	203.69596	No
	n00200	R01000	s000.kp	47836	17274	200.017915	No
	n00200	R10000	s000.kp	482118	174792	200.002795	No
	n00500	R01000	s000.kp	114616	42898	200.001716	No
	n00500	R10000	s000.kp	1155011	433675	200.001014	No
	n01000	R01000	s000.kp	228624	84656	200.000823	No
	n01000	R10000	s000.kp	2304627	856119	200.000893	No
<b>08SpannerWeaklyCorrelated</b>	n00050	R01000	s000.kp	10354	11452	7.563907	Yes
	n00050	R10000	s000.kp	102774	115687	5.244023	Yes
	n00100	R01000	s000.kp	20550	20824	0.051576	Yes
	n00100	R10000	s000.kp	203856	210158	0.049485	Yes
	n00200	R01000	s000.kp	40575	41116	25.454986	Yes
	n00200	R10000	s000.kp	403248	415714	152.06321	Yes

	n00500	R01000	s000.kp	98713	100076	200.065779	No
	n00500	R10000	s000.kp	981336	1011673	200.078885	No
	n01000	R01000	s000.kp	196050	198664	200.000395	No
	n01000	R10000	s000.kp	1948536	2008773	200.000752	No
<b>09SpannerStronglyCorrelated</b>	n00050	R01000	s000.kp	28440	11540	200.01082	No
	n00050	R10000	s000.kp	285753	116753	56.497167	Yes
	n00100	R01000	s000.kp	51656	20956	15.613069	Yes
	n00100	R10000	s000.kp	518835	211835	0.051946	Yes
	n00200	R01000	s000.kp	101888	41288	200.003268	No
	n00200	R10000	s000.kp	1022469	417469	200.000026	No
	n00500	R01000	s000.kp	245128	99928	200.038745	No
	n00500	R10000	s000.kp	2460672	1010672	200.053411	No
	n01000	R01000	s000.kp	488672	198772	200.00131	No
	n01000	R10000	s000.kp	4903566	2009566	200.000046	No
<b>10MultipleStronglyCorrelated</b>	n00050	R01000	s000.kp	21338	14238	0.00429	Yes
	n00050	R10000	s000.kp	217265	142265	0.000222	Yes
	n00100	R01000	s000.kp	43316	29016	2.28347	Yes
	n00100	R10000	s000.kp	434921	289921	0.000496	Yes
	n00200	R01000	s000.kp	81658	51558	31.155273	Yes
	n00200	R10000	s000.kp	813196	515196	7.408062	Yes
	n00500	R01000	s000.kp	203778	127278	200.001216	No
	n00500	R10000	s000.kp	2034682	1271682	200.000085	No
	n01000	R01000	s000.kp	399170	245970	200.000018	No
	n01000	R10000	s000.kp	3995528	2457528	200.00003	No
<b>11ProfitCeiling</b>	n00050	R01000	s000.kp	14229	14238	1.592672	Yes
	n00050	R10000	s000.kp	142272	142282	0.835323	Yes
	n00100	R01000	s000.kp	29001	29015	68.633119	Yes
	n00100	R10000	s000.kp	289947	289959	200.043842	No
	n00200	R01000	s000.kp	51540	51562	200.000048	No
	n00200	R10000	s000.kp	515169	515199	200.057079	No
	n00500	R01000	s000.kp	127239	127277	200.000393	No
	n00500	R10000	s000.kp	1271616	1271685	25.525246	Yes
	n01000	R01000	s000.kp	245877	245972	200.000051	No
	n01000	R10000	s000.kp	2457411	2457533	200.000029	No
<b>12Circle</b>	n00050	R01000	s000.kp	300031	14239	0.714557	Yes
	n00050	R10000	s000.kp	9485054	142283	4.750204	Yes
	n00100	R01000	s000.kp	611418	29017	0.21868	Yes
	n00100	R10000	s000.kp	19329757	289961	6.82233	Yes
	n00200	R01000	s000.kp	1086483	51563	200.074621	No
	n00200	R10000	s000.kp	34344928	515200	33.349096	Yes
	n00500	R01000	s000.kp	2681868	127278	200.000038	No
	n00500	R10000	s000.kp	84774716	1271685	200.000196	No
	n01000	R01000	s000.kp	5182856	245972	0.010647	Yes
	n01000	R10000	s000.kp	163827248	2457533	200.000992	No

*Bảng tổng hợp số lời giải tối ưu của các nhóm bài toán sử dụng OR-Tools*

Group Name	Not Optimal	Optimal	Total Tests
00Uncorrelated	0	10	10
01WeaklyCorrelated	0	10	10
02StronglyCorrelated	6	4	10
03InverseStronglyCorrelated	4	6	10
04AlmostStronglyCorrelated	5	5	10

05SubsetSum	0	10	10
06UncorrelatedWithSimilarWeights	2	8	10
07SpannerUncorrelated	8	2	10
08SpannerWeaklyCorrelated	4	6	10
09SpannerStronglyCorrelated	7	3	10
10MultipleStronglyCorrelated	4	6	10
11ProfitCeiling	6	4	10
12Circle	4	6	10

Theo kết quả từ bảng thống kê trên ta có thể nhận xét như sau:

- Cùng một loại instance, nhưng sự khác biệt trong size và range vẫn mang lại những kết quả lệch đáng kể (cùng size cùng range vẫn có sự lệch)
- Ở nhóm test case số 12 thì luôn có số lượng values lớn nhất so với các nhóm test case còn lại. Nhóm test case số 6 thì luôn có số lượng weight lớn nhất so với phần test case còn lại.

Để biết được nhóm nào dễ hay khó, em xem xét dựa vào số câu trả lời tối ưu:

- Các nhóm dễ giải quyết nhất với Google Or Tools là 00Uncorrelated, 01WeaklyCorrelated và 05SubsetSum.
- Các nhóm khó giải quyết nhất là 02StronglyCorrelated, 07SpannerUncorrelated, 09SpannerStronglyCorrelated và 11ProfitCeiling.