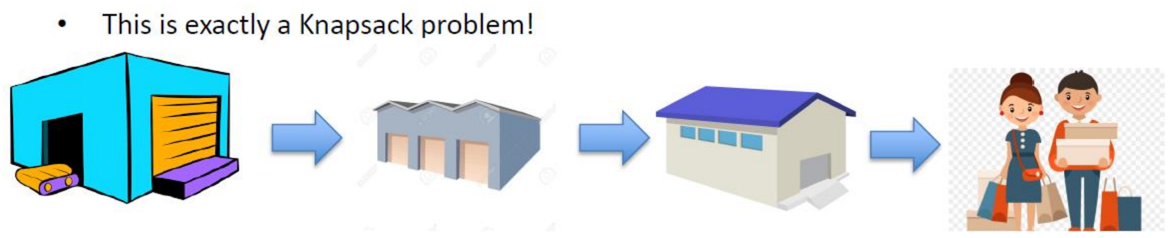
Midterm Project: Logistic Problem in Alibaba

Salih Tutun and Samira Fazel
Washington University in St. Louis



In the Alibaba company, we have one big warehouse, and customers need the products on time. Based on the coming online shopping, I need to send the products to them from the warehouse (seen in Figure 1). However, I have limited capacity in the warehouse and which products I need to store there. The question is here which products need to be in the warehouse before sending to customers. I will store and deliver my products dynamically.

This problem is an actual use case of the Knapsack Problem, and we store and deliver the products dynamically. We need to understand the knapsack problem and dynamic programming (shared as an extra file).

Knapsack Problem algorithm is a very helpful problem in combinatorics. In the warehouse there are n packages the package i has weight (Wi) and value (Vi). The company cannot store the products if the total products weights are greater than capacity. The problem to be solved here is: which packages will the company deliver to get the highest value?

Project Definition:

The Problem here we have 300 instances (trials). For each instance, we have C, Vi, and Wi for the problem.

Warehouse can store all boxes with C capacity

There are n items in the store.

Each item i: value Vi and a weight Wi

Capacity will be generated by using normal distribution as mean: 130 and standard deviation: 30. It means that we will generate the capacity between 100 and 160 for each instance. We shared all values and weights in the datasets for the project. For each instance, which item we need to select for getting more value and using less capacity.

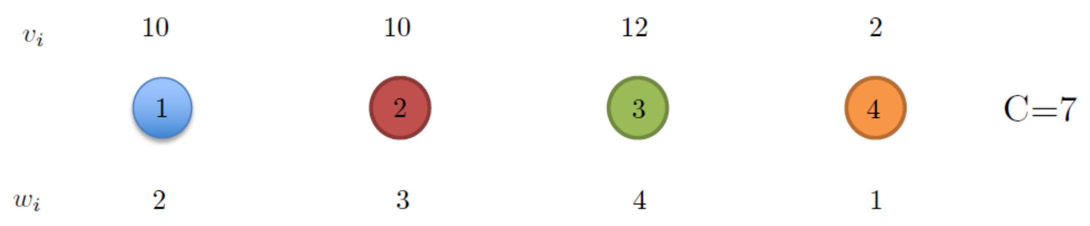
Which items should Alibaba put in Warehouse to maximize value?

Simple Examples for Knapsack Problem

Let me explain Knapsack problem with examples.

Note: Instance here is the trial or combinations of items.

For each knapsack instance,



For this example, we generated the capacity as 7 then we would like to optimize the total values.

We need to select the items that have minimum weights and maximum values. I will add each item dynamically and check my instance capacity.

When we select;

Item 1 (V:10, W:2) + Item 2 (V:10, W:3) + Item 4 (V:2, W:1) = Total V:22, W:6 so, W<C as 6<=7 with 22 values.

Item 2 (V:10, W:3) + Item 3 (V:12, W:4) = Total V:22, W:7 so, W<C as 7<=7 with 22 values.

Then, we will store [Item 1 and Item 2] or [Item 2 and Item 3] for this instance.

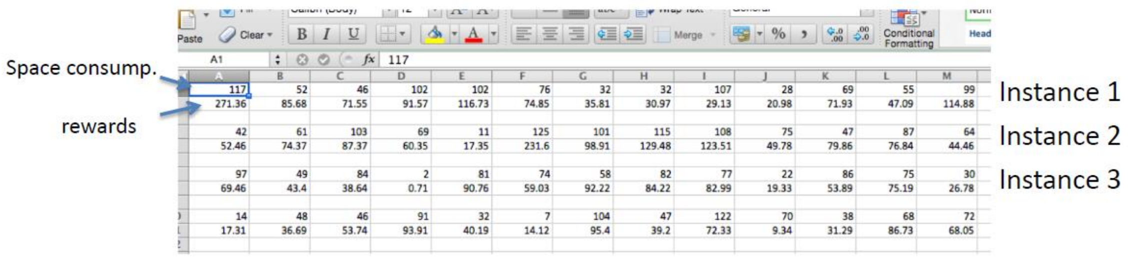
What we will do in the Project:

In the Project:

- 300 Warehouse (Knapsack) Instances
- For each instance, we need to define C in Normal Distribution[Mean:130, Standard Deviation:30]
- Only find out C once you choose which items to put in the warehouse (knapsack)
- If you go over C, then you get nothing for that instance

We generated 300 C for 300 instances. For example, let’s say that we generated 125 lbs, for Instance 1. It means that we store the items based on this capacity. As seen below, total space consumption cannot pass 125, for Instance 1.

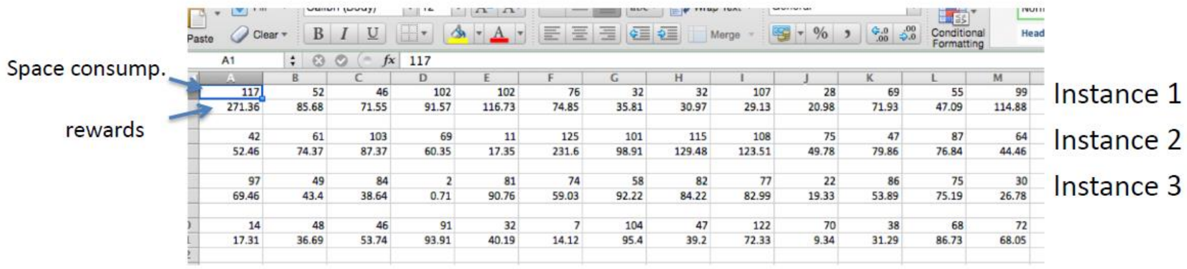
Here is how we’ll give you the knapsack instances:



A1 fx 117

A	B	C	D	E	F	G	H	I	J	K	L	M
117	52	46	102	102	76	32	32	107	28	69	55	99
271.36	85.68	71.55	91.57	116.73	74.85	35.81	30.97	29.13	20.98	71.93	47.09	114.88
42	61	103	69	11	125	101	115	108	75	47	87	64
52.46	74.37	87.37	60.35	17.35	231.6	98.91	129.48	123.51	49.78	79.86	76.84	44.46
97	49	84	2	81	74	58	82	77	22	86	75	30
69.46	43.4	38.64	0.71	90.76	59.03	92.22	84.22	82.99	19.33	53.89	75.19	26.78
14	48	46	91	32	7	104	47	122	70	38	68	72
17.31	36.69	53.74	93.91	40.19	14.12	95.4	39.2	72.33	9.34	31.29	86.73	68.05

<u>How to Submit the Project</u>

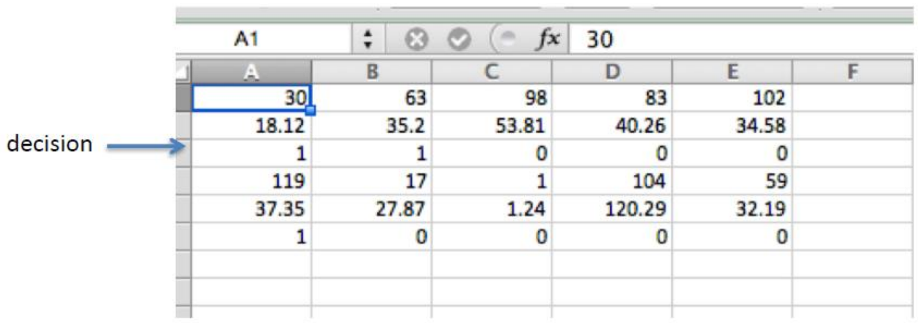


A1 fx 117

A	B	C	D	E	F	G	H	I	J	K	L	M
117	52	46	102	102	76	32	32	107	28	69	55	99
271.36	85.68	71.55	91.57	116.73	74.85	35.81	30.97	29.13	20.98	71.93	47.09	114.88
42	61	103	69	11	125	101	115	108	75	47	87	64
52.46	74.37	87.37	60.35	17.35	231.6	98.91	129.48	123.51	49.78	79.86	76.84	44.46
97	49	84	2	81	74	58	82	77	22	86	75	30
69.46	43.4	38.64	0.71	90.76	59.03	92.22	84.22	82.99	19.33	53.89	75.19	26.78
14	48	46	91	32	7	104	47	122	70	38	68	72
17.31	36.69	53.74	93.91	40.19	14.12	95.4	39.2	72.33	9.34	31.29	86.73	68.05

You need to define 0 or 1 in this CSV file. 1 means that Alibaba do store this item on the warehouse as seen below. For example, for instance 1, we selected item 1 and item 2 and total space consumption is 30+63=93. This 93 is less than 125. As a reward, we earned 18.12 + 35.2 dollars. So, we need to assign these 0s and 1s for earning more by using less capacity.

Turn in your Answer as “Name.csv”



A1 fx 30

A	B	C	D	E	F
30	63	98	83	102	
18.12	35.2	53.81	40.26	34.58	
1	1	0	0	0	
119	17	1	104	59	
37.35	27.87	1.24	120.29	32.19	
1	0	0	0	0	

In the evaluation.ipynb file, you will show that how much you are earned. Moreover, you will show all assignments in this CSV files for submission.

After defined all packages (items) to warehouse, we need to find the optimum way by using monte carlo simulation. It means that we will use the average of all values for showing.

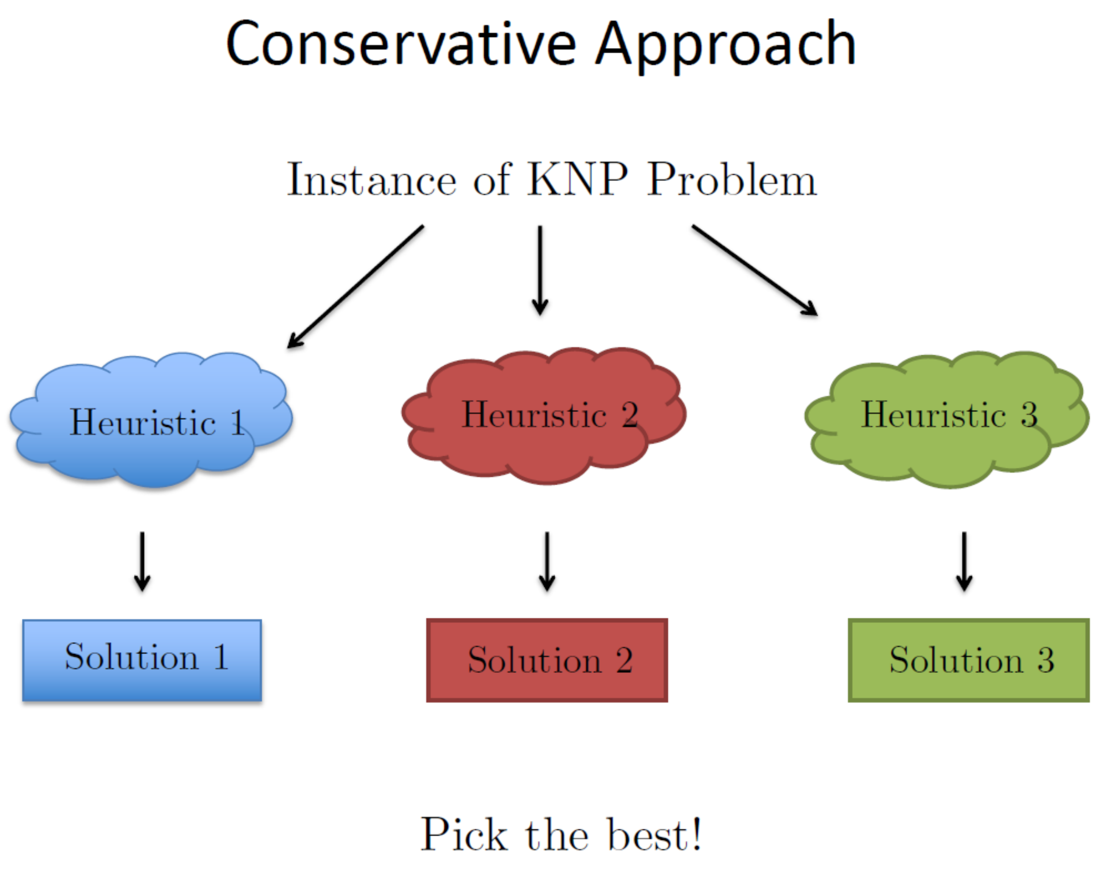
Conservative Approach

Assume the size of the knapsack is 100 for all instances

- This way you will never go over

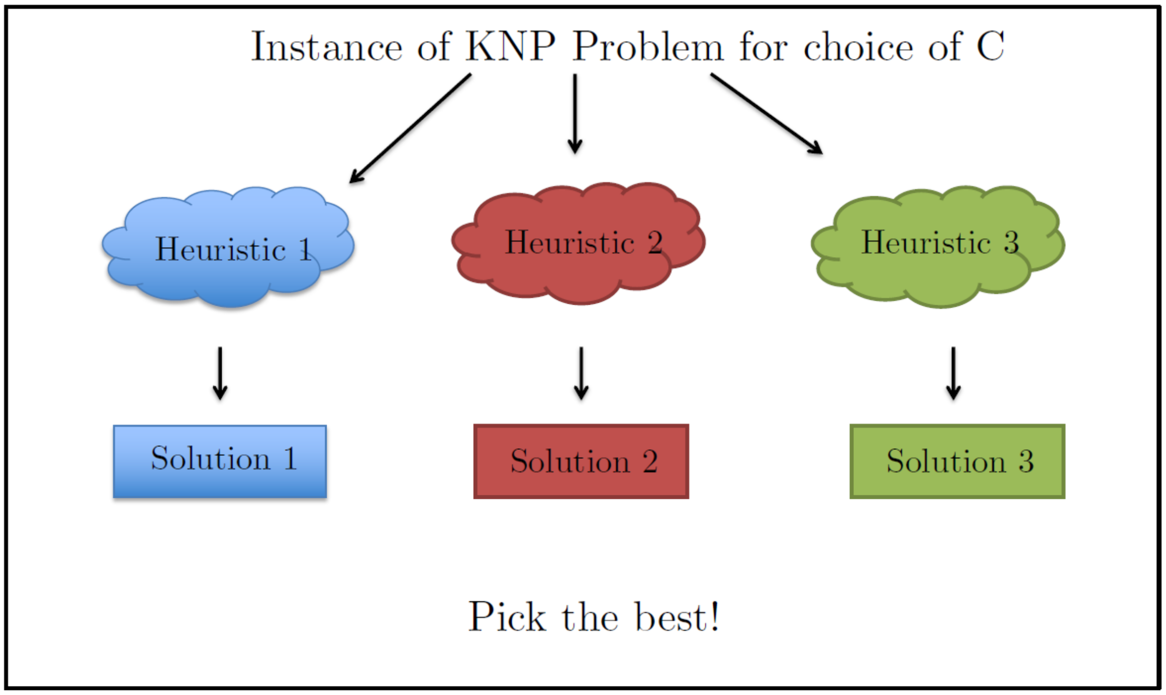
• Find best way to pack knapsack.

- Develop a few heuristics.
- For each instance of the knapsack problem, pass the instance to each heuristic and then pick the best.



If I assume all knapsacks have size 100, I might be leaving points on the table.

- Use Monte Carlo to find a good mix of knapsack sizes.
- Start by assuming all knapsacks are the same size.



Monte Carlo Simulation

Output: In this project, we need to show the best way to pack the warehouse (knapsack). For example, we have 300 instances (trials). Please show which instance is the best and how much we will earn.

Note: You need to use Monte Carlo Simulation for this part.

Grading

• 27 points total

7 points towards how good your solution is

- Top Three Place (among all groups across sessions) – 7 points
- Rest of top 33% - 5 points
- Middle 33% - 3 points
- Bottom 33% - 2 point

20 points towards the fact your code actually runs.

- Your code needs to pass the auto-grader and provides a nonnegative value.