The Knapsack Problem

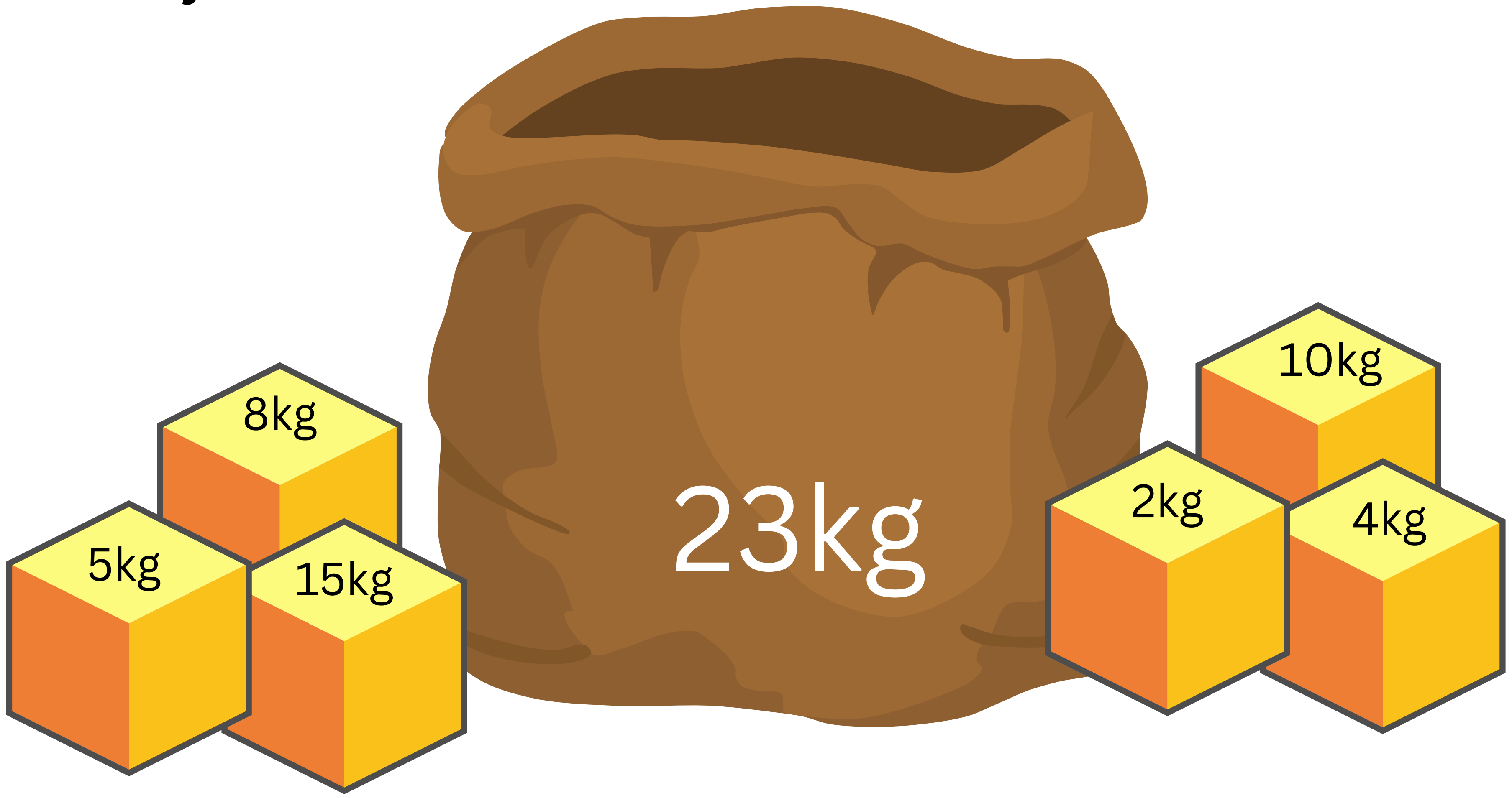
Team 2

Radhika Khatri
Ali Asghar Chakera
Abbas Taqvi
Haider Abbas
Aumaima Rahid

What is the Knapsack Problem?

Input:
No. of items - **N**
Weight/Capacity -**W**
profit - **pi of every item i**
weight - **wi of every item i**

Output:
List of selected items
Max profit



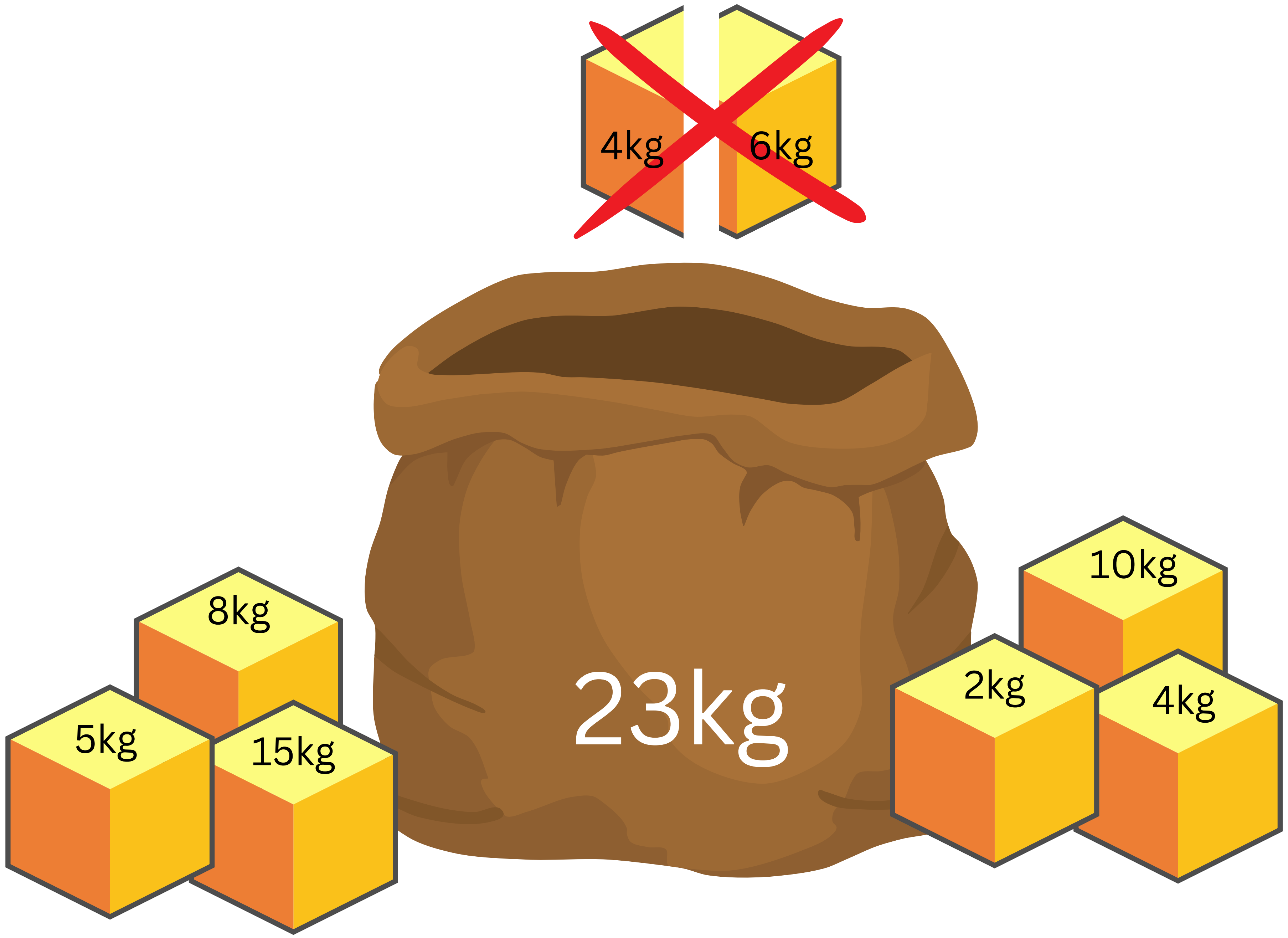
The name "knapsack problem" dates back to the early works of the mathematician Tobias Dantzig (1884–1956), and refers to the commonplace optimization problem of packing the most valuable or useful items without overloading the luggage.

Our Agenda

The constraint here is we can either put an item completely into the bag or cannot put it at all [It is not possible to put a part of an item into the bag].

There are three types of knapsack problems :

- **0-1 Knapsack** ✓
- Fractional Knapsack
- Unbounded Knapsack



Brute Force

item	a	b	c	profit,weight
approach 0	0	0	0	0, 0
approach 1	0	0	1	8, 4
approach 2	0	1	0	3, 1
approach 3	0	1	1	11, 5
approach 4	1	0	0	12, 3
approach 5	1	0	1	20, 7
approach 6	1	1	0	15, 4 ✓
approach 7	1	1	1	23, 8

pi/wi ratios

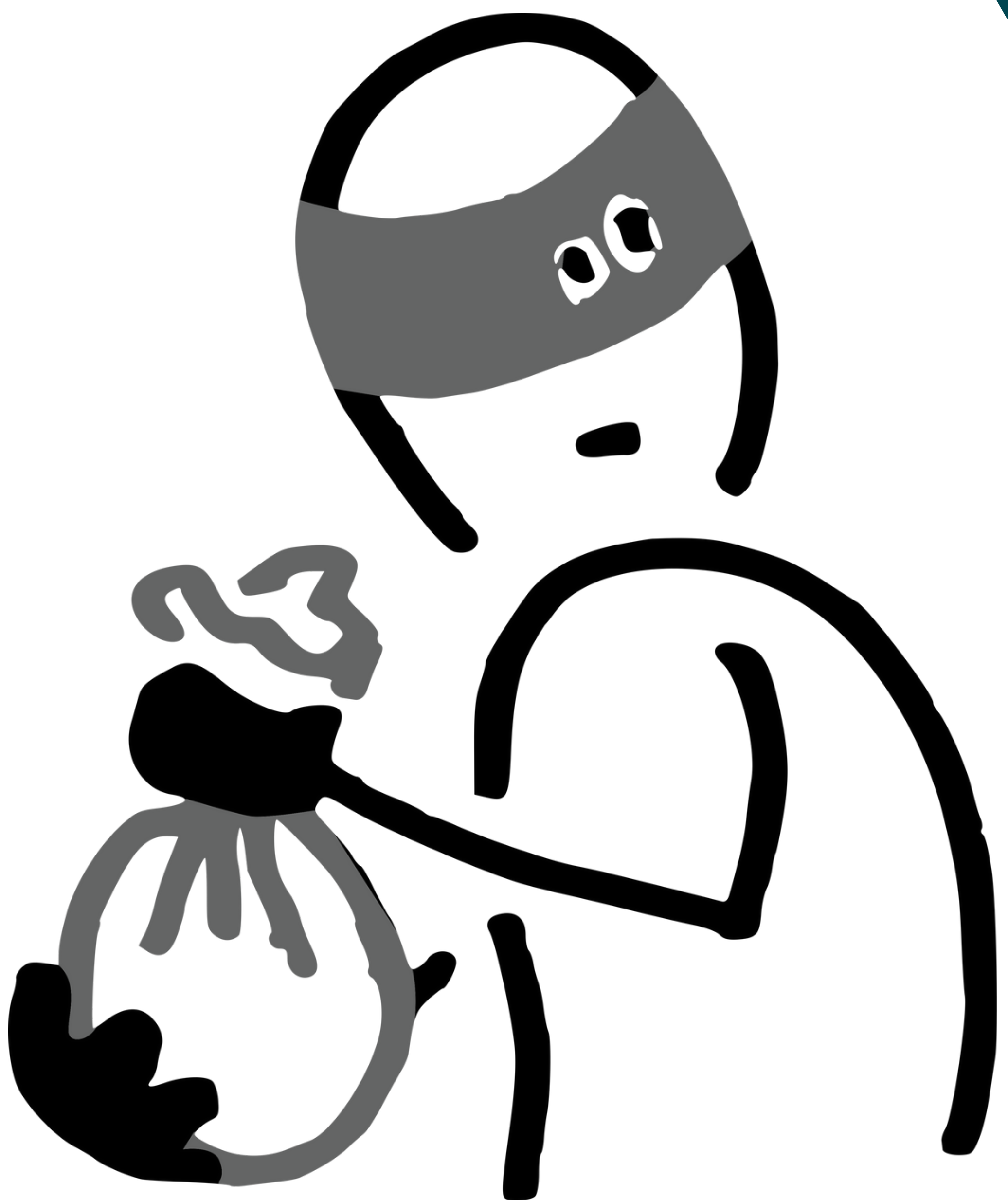
a = 12/3

b = 3/1

c = 8/4

Weight

6



Approaches & Implementations

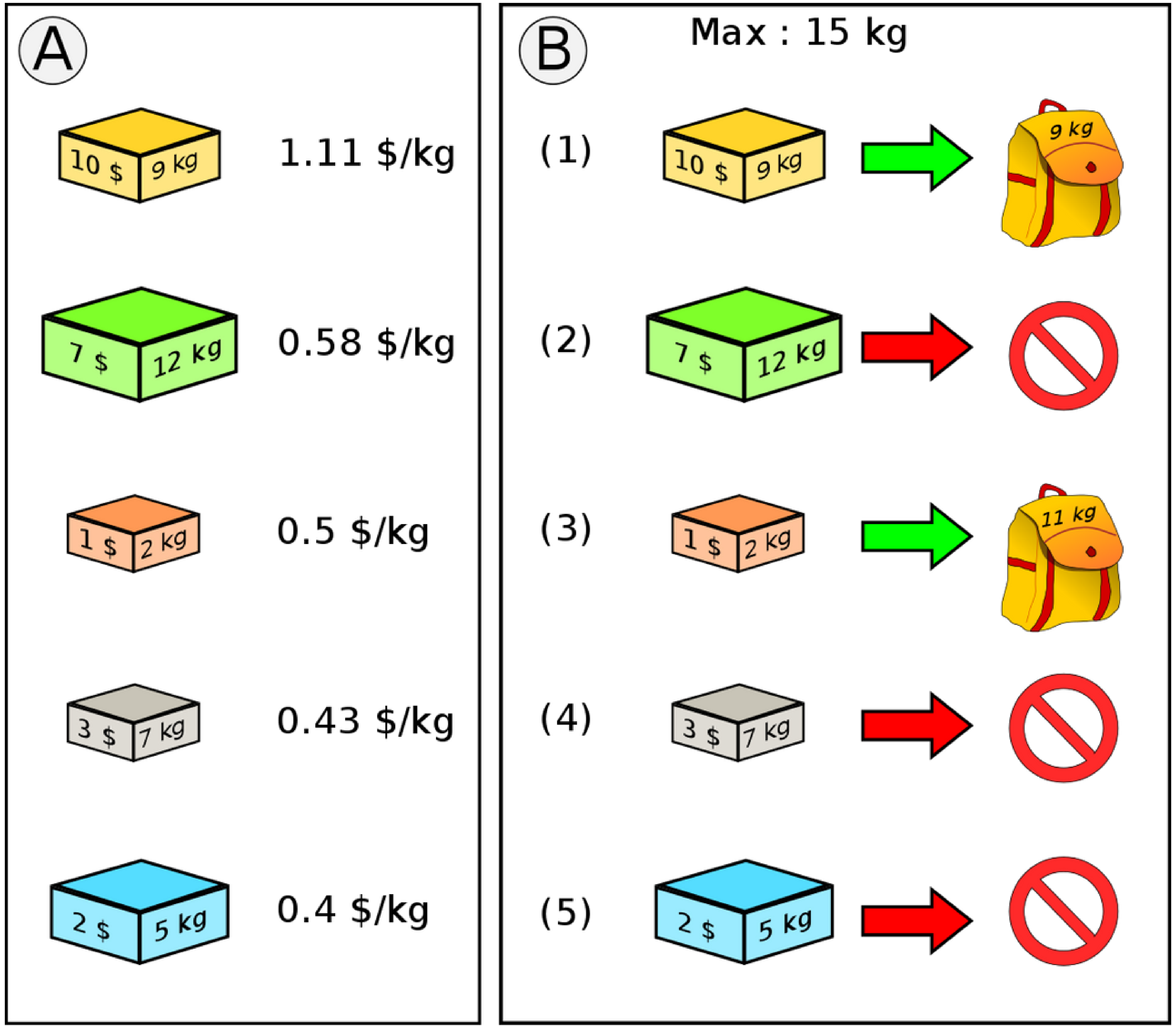
Greedy Approach
O(n log n)

Dynamic Programing
O((n)(wmax))

Evolutionary Algorithm
O(gpn(p + 1))

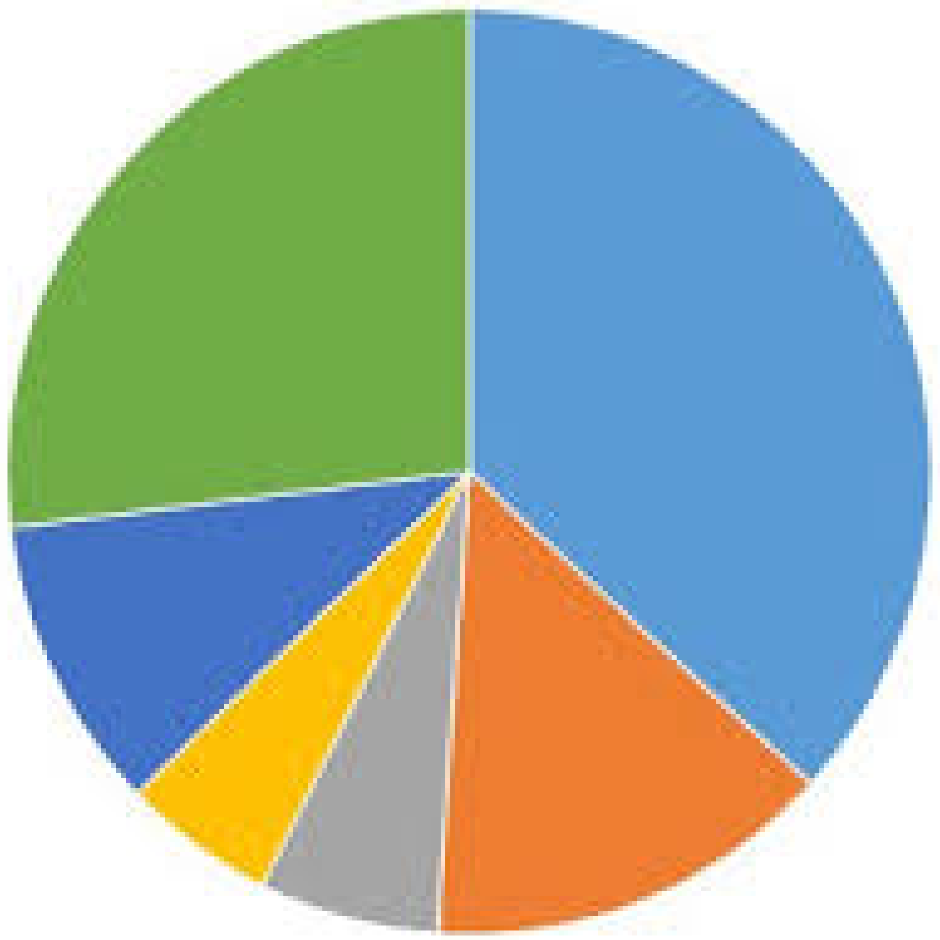
Greedy Approach (O(nlogn))

- Arrange items in decreasing order of weight to value ratios.
- Pick the items that do not exceed the weight capacity.
- Subtract items from total weight capacity, repeat till **pi/wi > W**
- Return the set of selected items and the total value.



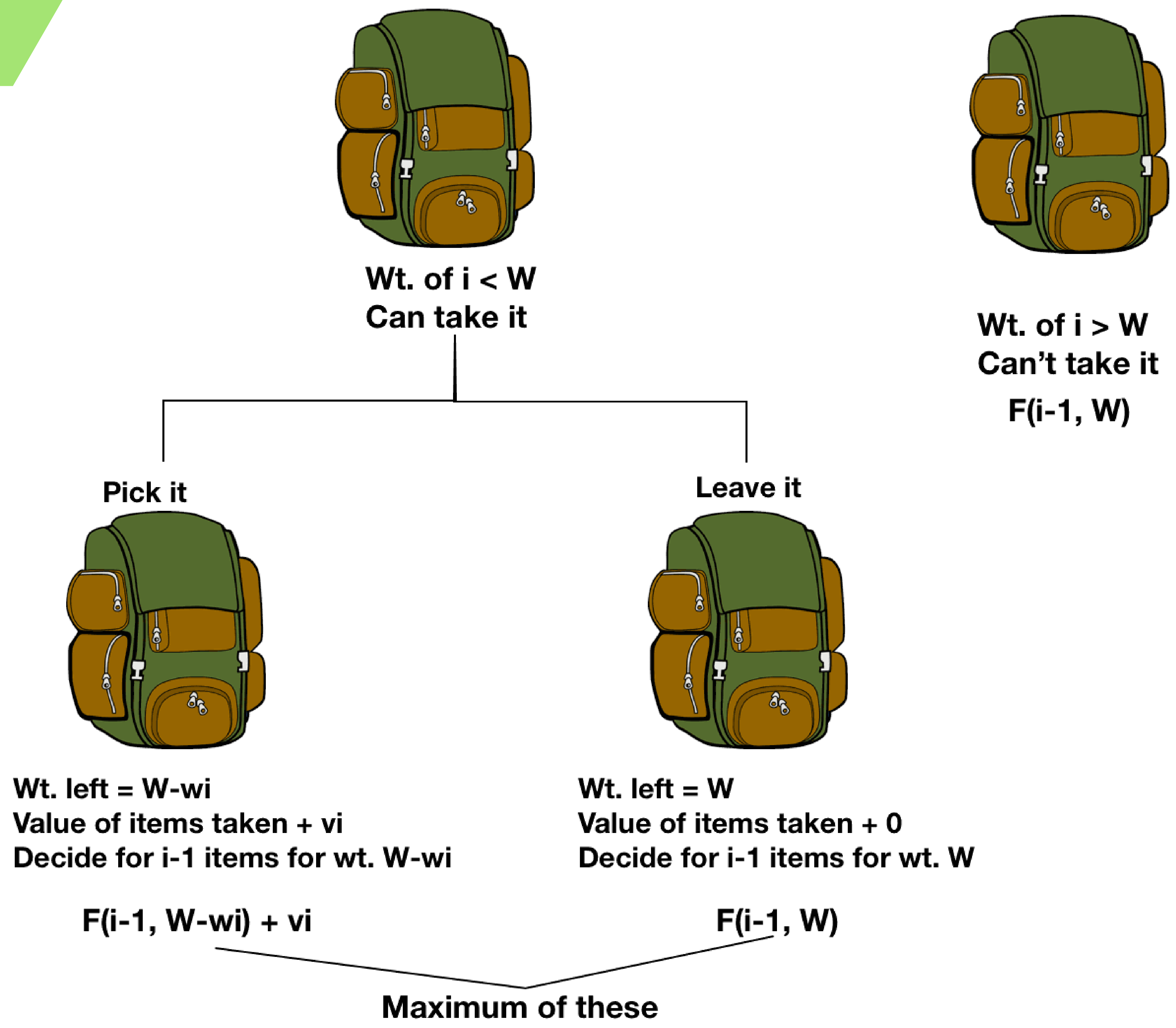
Evolutionary Algorithm O(gpn(p+1))

- Generate P random solutions: Population
- Calculate the fitness of the population
- Select 2 parents through binary tournament
- Generate offspring through Crossover
- Mutate the offspring
- Select P individuals through rank based selection for next generation



Dynamic Programming
Θ(nW)

- Find solutions of the smallest subproblems.
- Find out the formula (or rule) to build a solution of subproblem through solutions of even smallest subproblems.
- Create a table that stores the solutions of subproblems. Then calculate the solution of subproblem according to the found formula and save to the table.
- From the solved subproblems, you find the solution of the original problem.



Datasets

In each dataset instance:

- First line should have the optimum.
- The next line contains space separated n and wmax
- The n lines following them contains space separated vi and wi where i ranges from 0 to n-1.

A script parses any instanceby taking the filepath and returning a list containing [optimum, n, wmax, V, W]

01 Base Dataset (By artemisa)

02 Very Large n (Number of items)

03 Very Large wmax (Capacity)

04 Very large n & wmax

05 Very large values of vi (Profit)

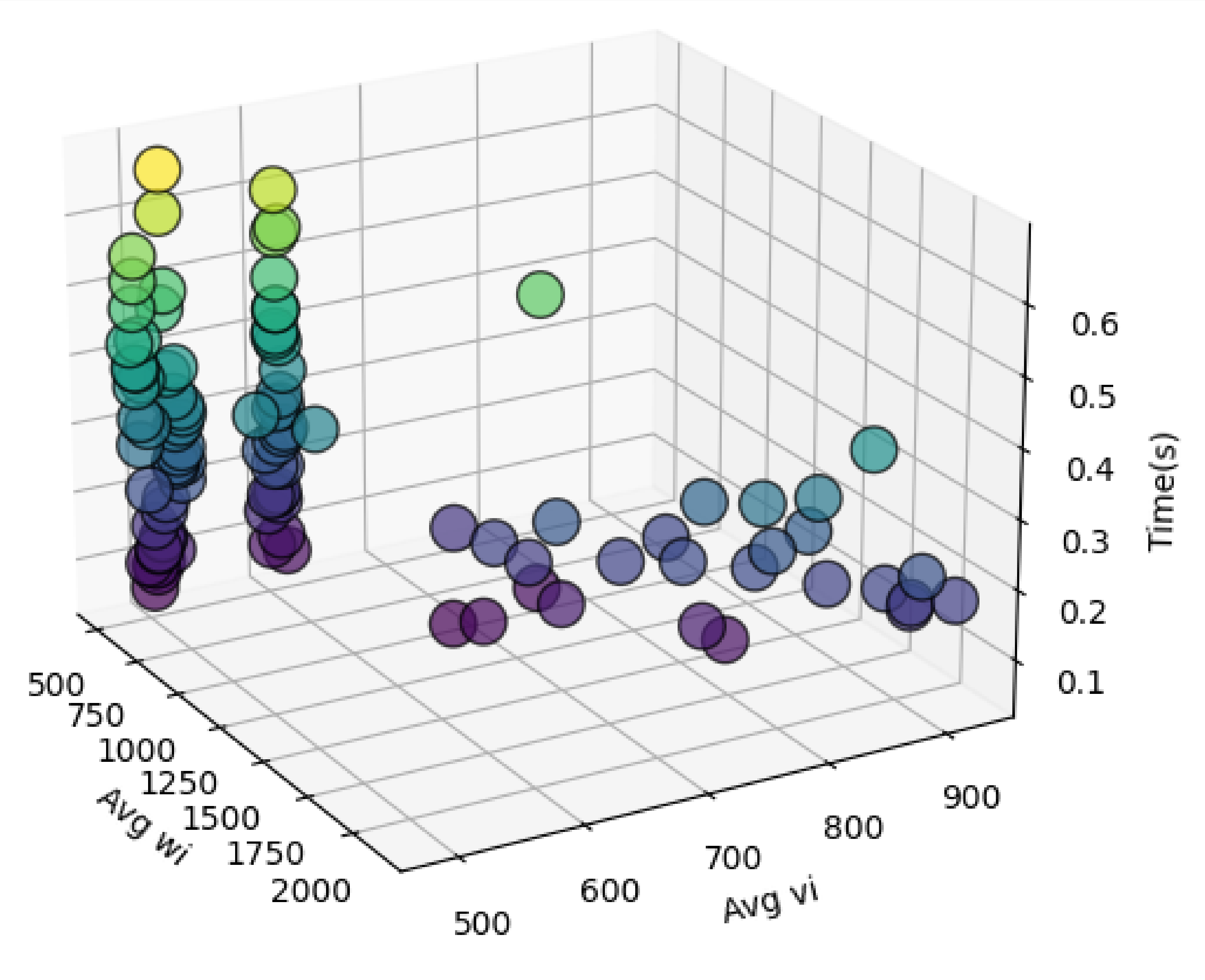
06 Very large values of wi (Weight)

07 Very Large values of vi & wi

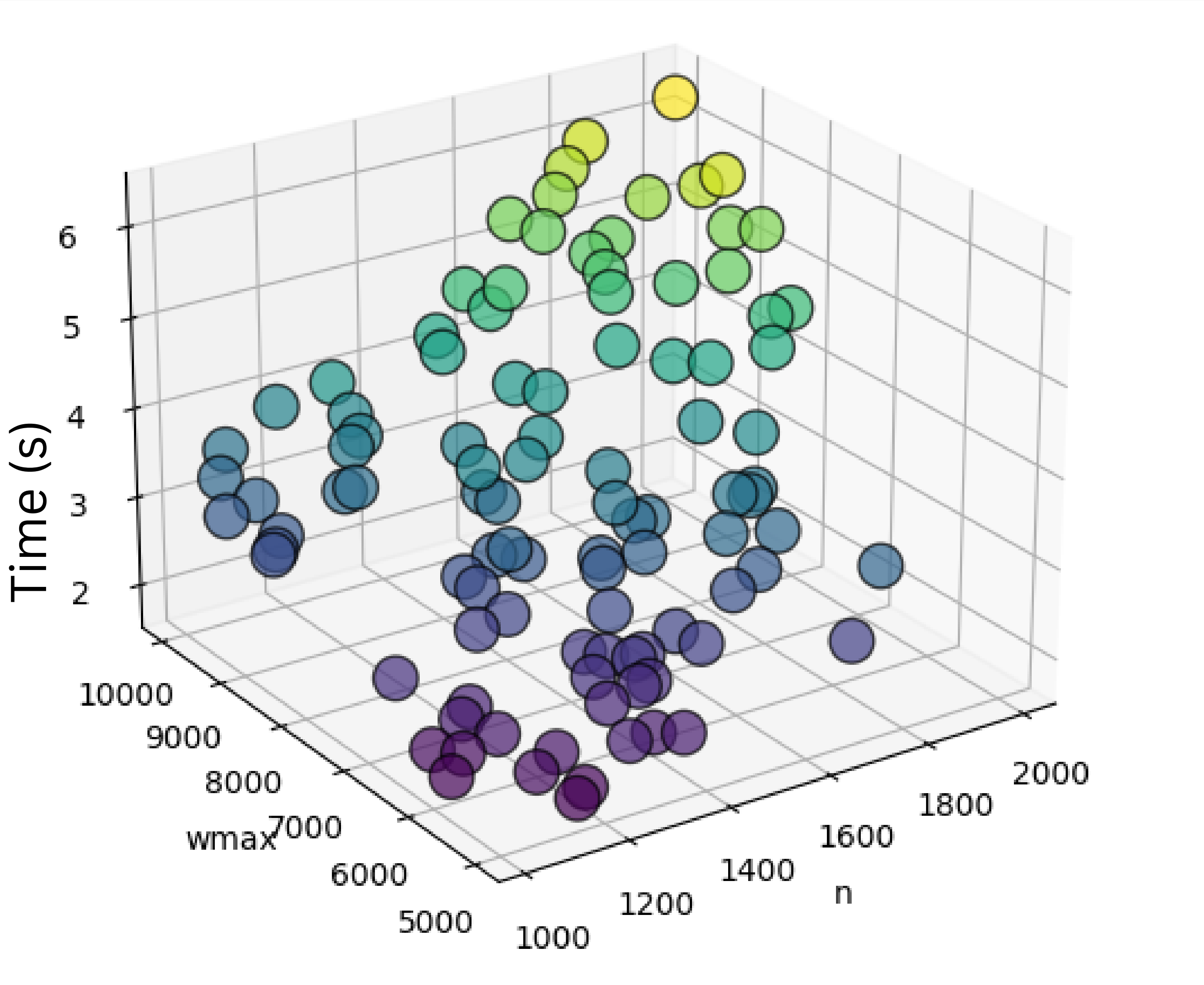
Analyzing Our Approaches

DP

very large vi and wi



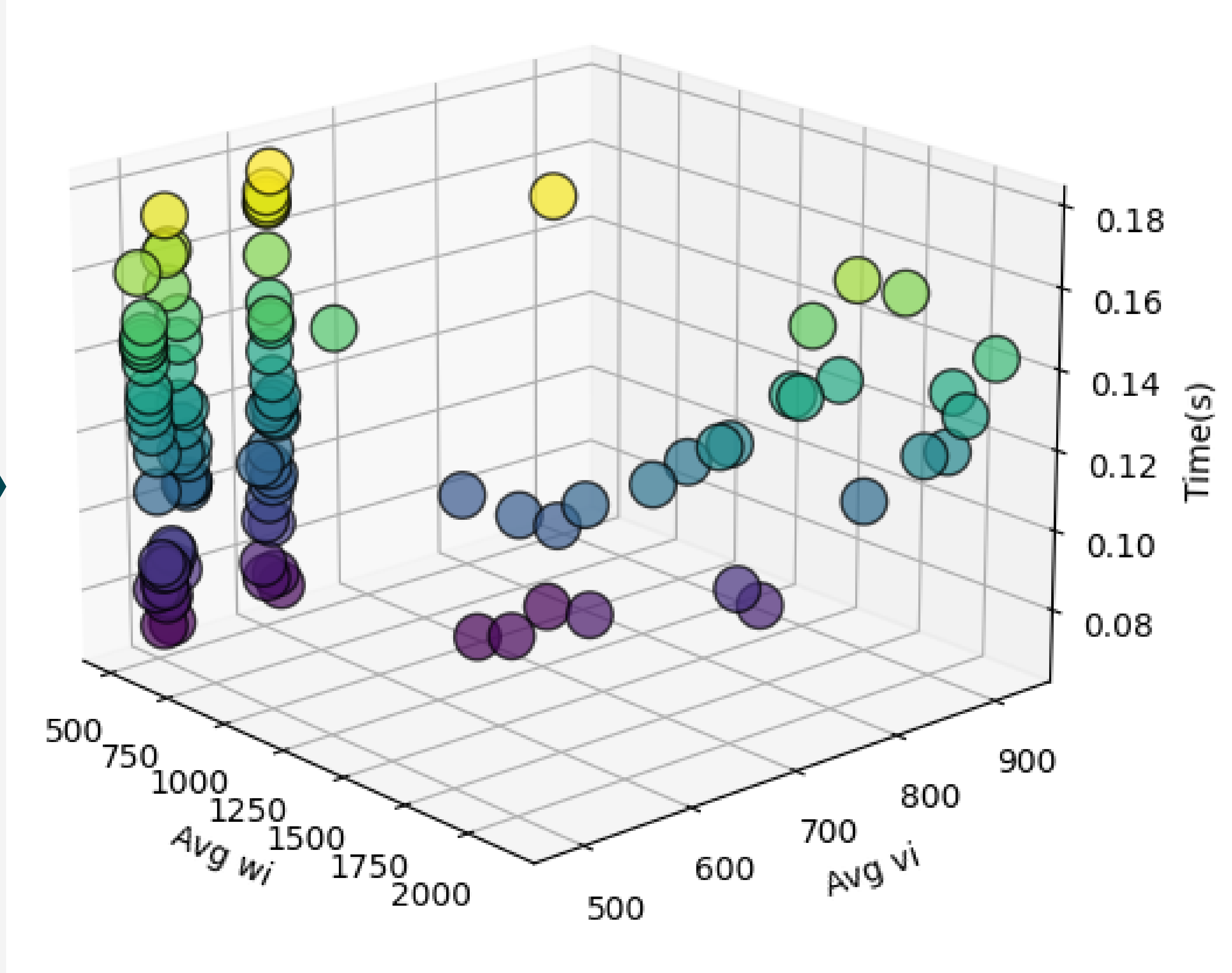
very large n and wmax



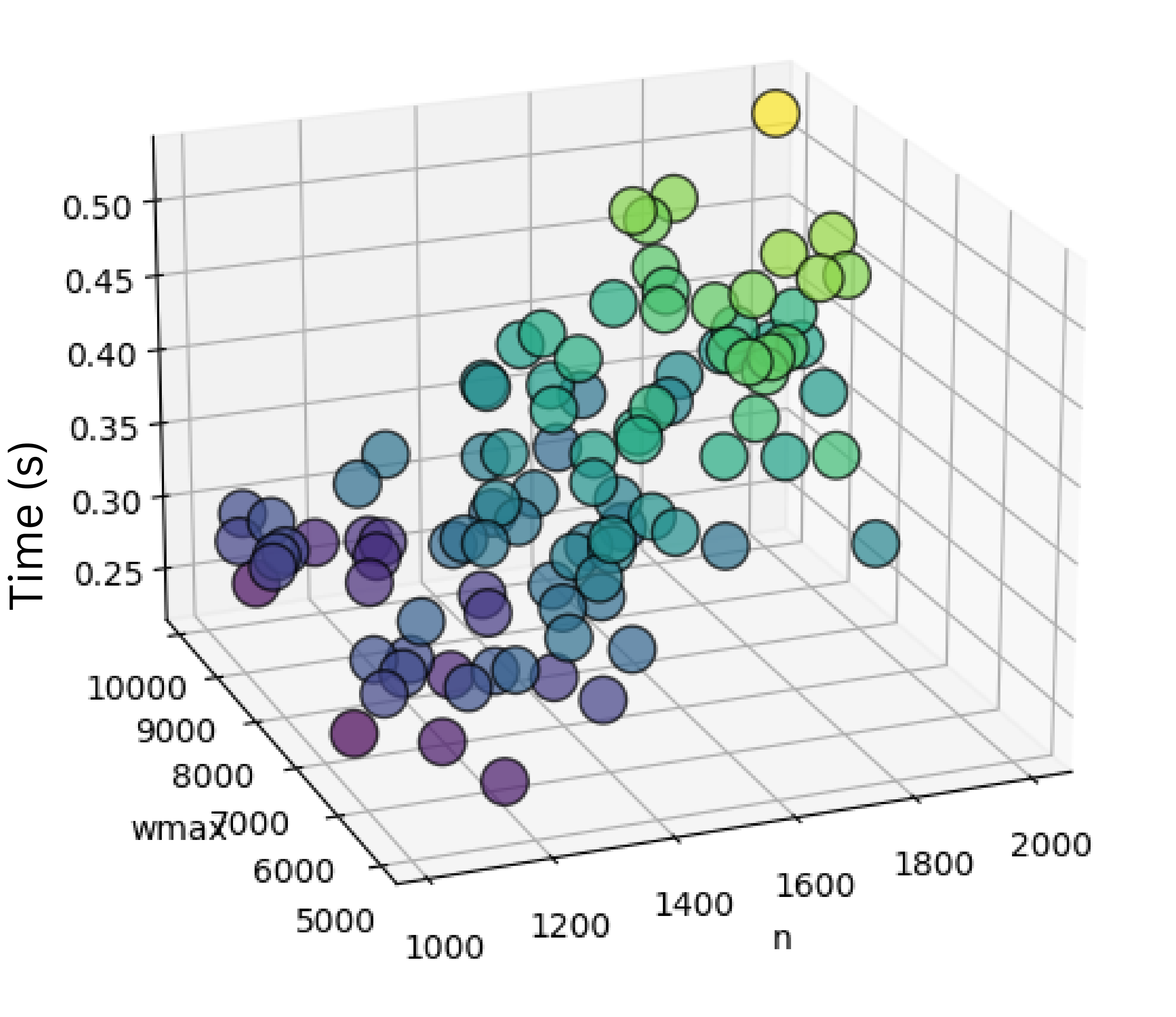
Analyzing Our Approaches

EA

very large vi and wi



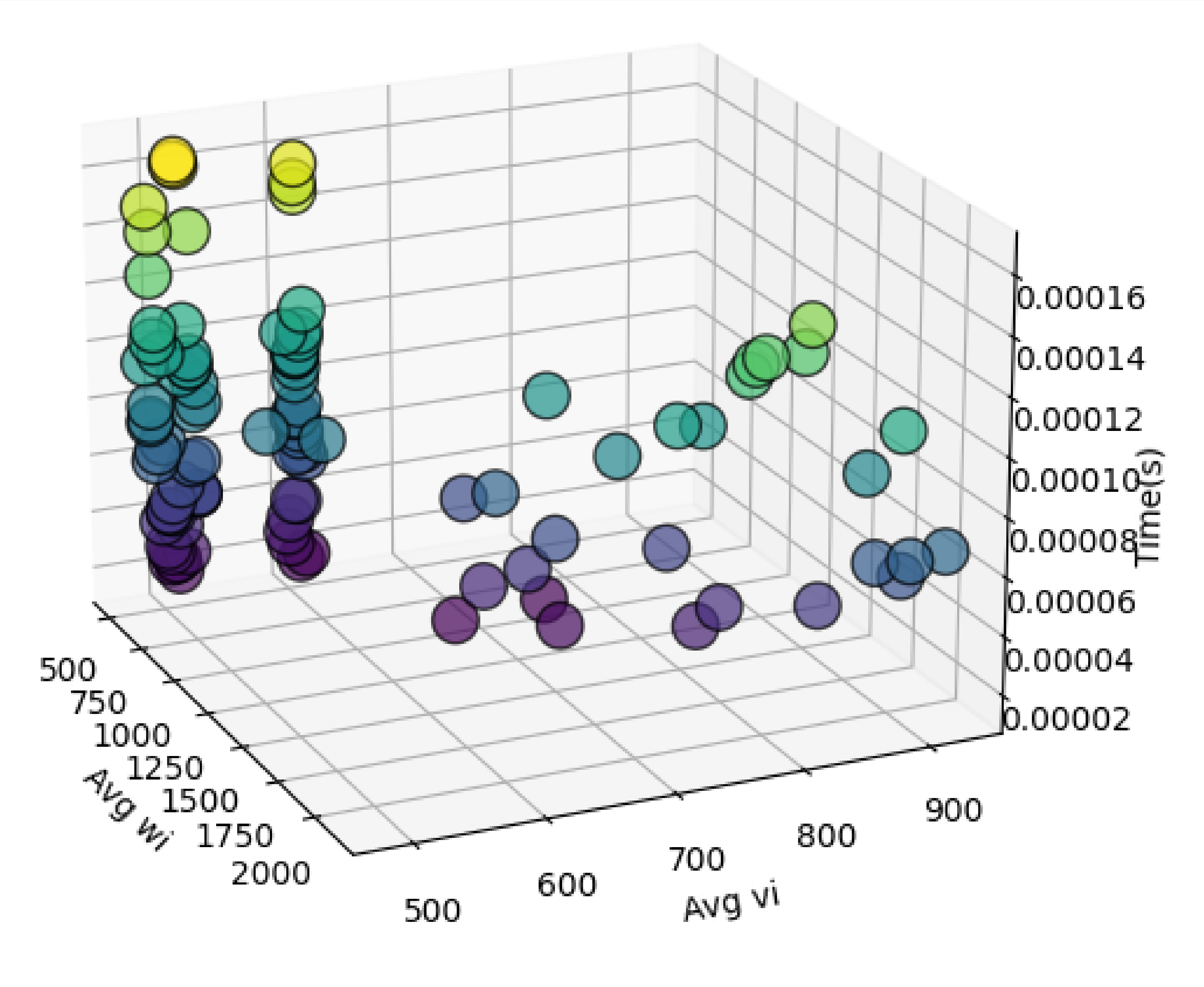
very large n and wmax



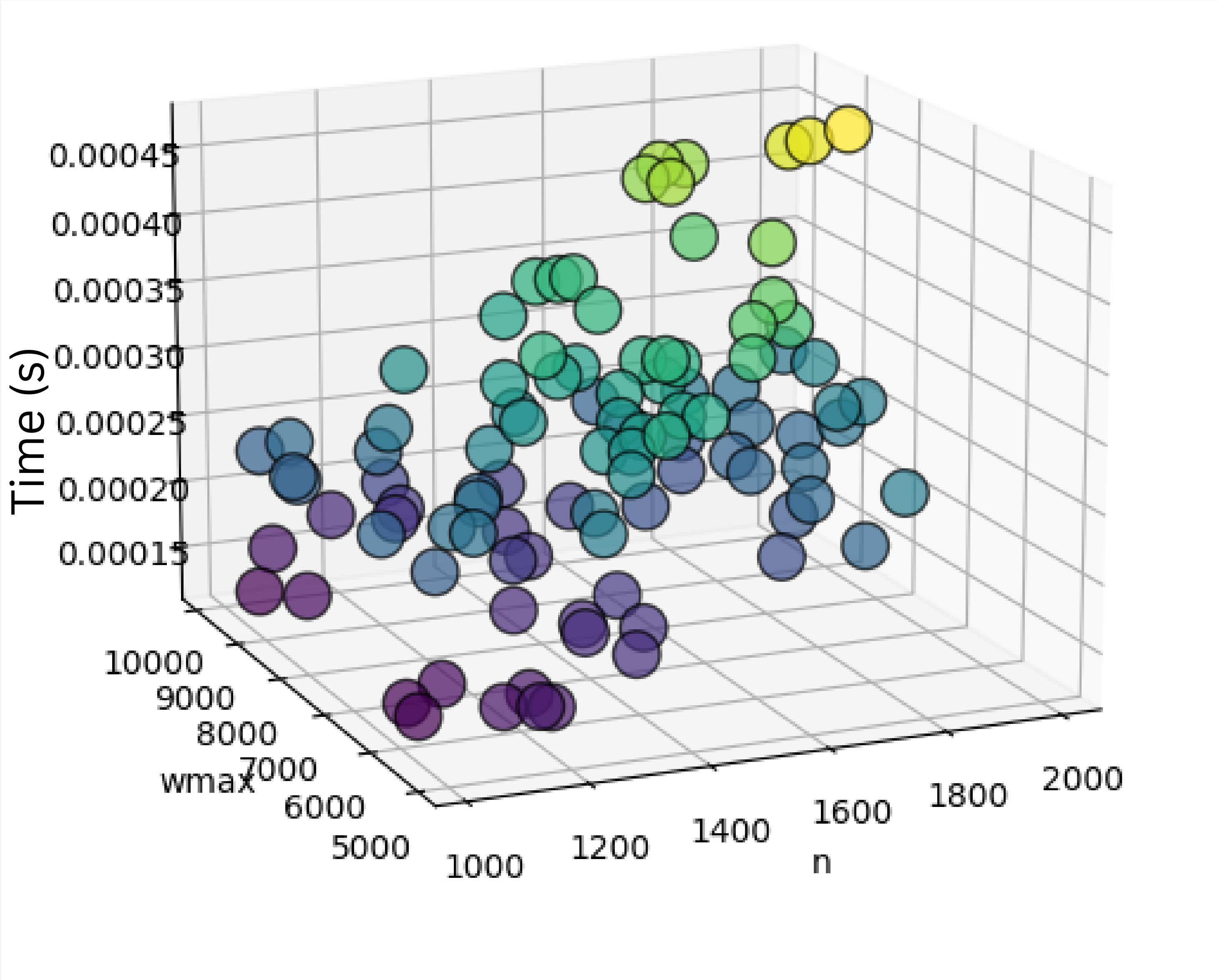
Analyzing Our Approaches

Greedy

very large vi and wi



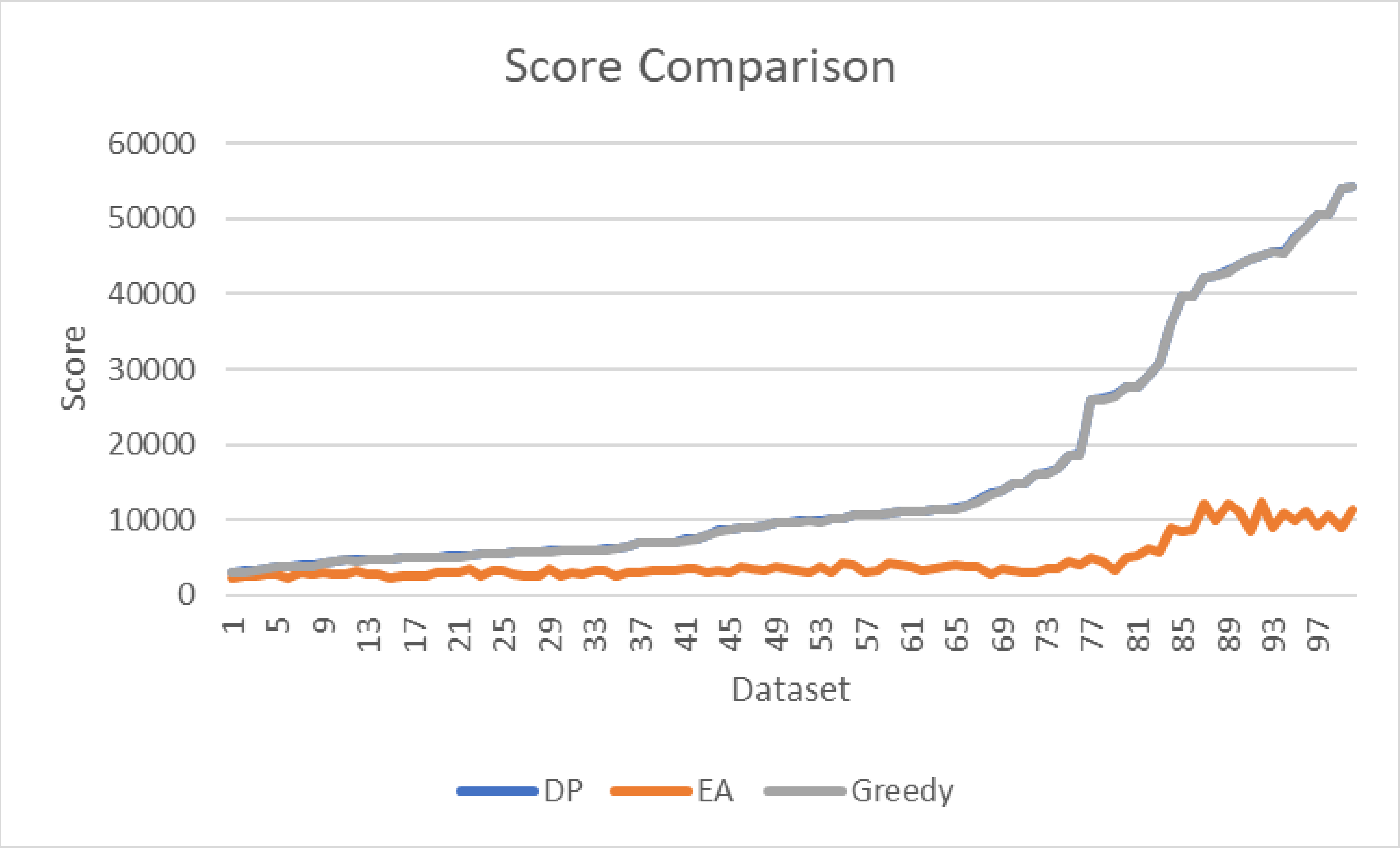
very large n and wmax



Analyzing Our Approaches

EA Error (%) = 60.65633271

Greedy Error (%) = 0.530543669



Greedy is overlapping with DP

Thankyou, Questions?

References

Evolutionary Algorithm:
https://github.com/sa06840/CI-Homework-1.git

Dynamic Programming:
https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/

Greedy Approach:
https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/

GitHub Link:
https://github.com/aliasgharchakera/DAA-Spring23-Project